UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Protección de la privacidad en la gestión de eventos de calendario en la nube

Máster Universitario en Ingeniería Informática

Autor: Pico Paredes, Rodrigo Xavier
Tutor: Arroyo Guardeño, David

Julio 2017

Agradecimientos

Con las siguientes palabras me gustaría agradecer a todas las personas que han formado parte y me han acompañado durante el largo camino que he recorrido hasta este momento. Sin ellas, en mayor o en menor medida, no hubiera sido capaz de superar algunas de las dificultades que siempre se presentan, y no hubiera conseguido alcanzar las metas que me he ido marcando.

Tengo que hacer especial referencia a algunas personas que me han ayudado bastante y me han hecho más amena esta etapa final de mi trayectoria educativa.

Varios amigos con los que he compartido carrera, incluso clases y prácticas, y los cuáles ya conocía desde el instituto. Nuevos amigos que he hecho durante mi paso por la Universidad, tanto durante el grado como ahora en el máster. A David, por ser un gran profesor/tutor, pero sobretodo una gran persona, que me ha guiado y ha creído en mí durante la realización de mis proyectos, haciéndose partícipe como si fueran suyos. A mi familia, pero en particular a mis padres por todo lo que hacen para que pueda cumplir mis objetivos.

A todos y a cada uno, y sin querer dar casi nombres, porque luego siempre falta alguien por mencionar, muchas gracias de verdad. =D

Resumen

Hoy en día muchos de los recursos informáticos que se emplean pueden ser ofrecidos como servicios. Así, muchas son las empresas que han lanzado al mercado diversas aplicaciones y herramientas en forma de servicios *cloud*. La facilidad de acceso a los sistemas en la nube a través de Internet ha permitido a los usuarios finales realizar todo tipo de tareas cotidianas, tanto personales como laborables. No obstante, estas rutinas suponen, *de facto*, una externalización de la custodia y gestión de la información, lo que ha originado un intenso debate en la sociedad en torno a la seguridad y privacidad de los datos en la nube. En efecto, no siempre es posible saber con claridad la forma de administración ni la finalidad de algunas de las operaciones que los proveedores *cloud* realizan con nuestros datos. No basta con que dichos proveedores establezcan medidas de protección frente a agentes externos maliciosos. El carácter central de la información exige ir un paso más allá desplegando de modo transparente soluciones que permitan conseguir un adecuado equilibrio entre seguridad y privacidad.

Las aplicaciones *cloud* son muy variadas, y entre sus servicios se encuentran los gestores de eventos de calendario. Estos productos son cada vez más demandados, sobretodo en el ámbito profesional, dado que permiten organizar y planificar las tareas en un marco temporal concreto. Con este panorama y con las inquietudes antes mencionadas, se decidió crear una aplicación *software* para gestionar de modo seguro y privado calendarios almacenados en la nube. Asimismo, y con objeto de garantizar la disponibilidad del servicio, se concibió la aplicación como una solución de escritorio con la posibilidad de modo de ejecución *offline*, esto es, que puede utilizarse sin conexión a Internet. Además, la aplicación es multiplataforma y *OpenSource*, para proporcionar más diversidad en los entornos de utilización, y para dar a la comunidad y a la sociedad una herramienta de *software* libre que, consecuentemente, pueda ser analizada y auditada para saber qué hace y cómo lo hace.

Palabras Claves

Gestión de eventos de calendario, nube, cloud computing, seguridad de la información, privacidad, aplicaciones de escritorio multiplataforma.

Abstract

Today many of the former computer-based functionality can be interpreted as cloud services. As matter of fact, the ease to access Internet and the quality of the service have fostered the adoption of cloud applications by more and more users. Consequently, it is possible to assert that the cloud is a central hub of many of our daily personal and professional activities. Nevertheless, we should take into account that the storage and processing of information in the cloud entail the outsourcing of the data custody. Indeed, we trust cloud service providers to protect our information and to access it only upon our informed consent. These assumptions are not always satisfied, which has pave the way for an intense debate around the difficult tradeoff between security and privacy in cloud services.

Among all the possible cloud applications, in this work the focus is on cloud-based event managers. These applications are increasingly used, especially in the professional field, since they allow to organize and plan tasks in a very convenient way. On the grounds of the aforementioned security and privacy threats, this project was intended to design and implement a secure and privacy-respectful software application to handle cloud-based calendars. Service availability was also a major concern in this project, and thus the application was designed to enable offline access to calendar events. Finally, the application was conceived as a cross-platform and *OpenSource* desktop solution, which further endorses the service availability and auditability. Certainly, the resulting product can be used in different operating systems and its source code is publicly available to be thoroughly analyzed and improved.

Key Words

Calendar-based event management, cloud, cloud computing, information security, privacy, security-by-design, privacy-by-design, cross-platform desktop applications.

Índice de Contenidos

Índice de Figuras

1. Introducción

Este trabajo se presenta con la finalidad de estudiar **la seguridad y la privacidad en los entornos *cloud* orientados a los servicios**, y más concretamente, en las **aplicaciones de gestión de eventos de calendario**. Además, se incluye el desarrollo de una aplicación de este estilo, que sirva de ejemplo y apoye los propósitos iniciales perseguidos. Este proyecto es una extensión y cambio de enfoque en algunas características del trabajo realizado en el grado de ingeniería informática [1].

En este apartado se van a definir diferentes aspectos relevantes vinculados con la inicialización y el transcurso del presente proyecto. Estos contenidos se verán posteriormente reflejados en las motivaciones, y serán desarrollados a través de los objetivos principales del actual proyecto. Una vez expuestos los hitos a cumplir y las necesidades específicas del proyecto, se detalla la planificación y la organización temporal para llevar a cabo las tareas asociadas. Finalmente, se describe la estructura general del documento.

1.1. Motivación

En las últimas décadas, el desarrollo de las **Tecnologías de la Información (TI)** y la informática han evolucionado de forma vertiginosa, pero no siempre lo ha hecho de forma solidaria con una correcta **protección de la seguridad y privacidad** de los datos de los usuarios de estas tecnologías. Esto ha dado lugar a la creación de entornos incompletos para su uso por los usuarios finales, lo que conlleva suministrar sistemas frecuentemente vulnerables. Esto es de especial relevancia debido a que **la actividad diaria de nuestra sociedad depende cada vez más de las tecnologías de la información** y, por tanto, sería recomendable tomar en consideración los riesgos asociados al uso de determinadas tecnologías en los diversos contextos de aplicación.

El auge de muchos elementos relacionados con estas tecnologías ha generado una dependencia del ciudadano respecto a las TI. Nuestro modelo de interacción social y de actividad económica está claramente condicionado por ellas, de modo que parece necesario perfeccionarlas y protegerlas [2]. Ahora bien, la **correcta protección de la información** sólo es factible contando con la

participación activa de los usuarios. Es por ello que es crítico educar al conjunto de usuarios TI con los temas relacionados en torno a la seguridad y privacidad de su información. En un contexto en el que el ciudadano está casi siempre conectado a Internet, éste ha de saber que en la mayor parte de los casos **se delega la custodia de sus datos en terceros**. En determinadas circunstancias en las que entran en juego agentes externos maliciosos [3] o terceras partes negligentes [4], esto puede implicar que parte de esa información que se deposita en Internet sea empleada en contra de su propietario, o que dicho propietario deje de tener acceso a su información. Este riesgo es **especialmente relevante cuando se hace uso de la denominada nube**.

Los servicios *cloud* son herramientas que proporcionan entornos de **almacenamiento y procesamiento de la información de manera ubicua y con diversas modalidades de acceso** (la nube como *software*, la nube como plataforma, y la nube como infraestructura). Estas características son de importancia en diferentes contextos, público, privado o educativo, y en distintos elementos, coches, móviles, ordenadores, televisores. En cada uno de dichos modos, el usuario deposita información, por lo que está confiando en el proveedor de servicios como protector de la seguridad y de la privacidad de sus datos. El hecho que existan **distintos niveles de confianza** en cada modelo de acceso no es en sí una amenaza. Ahora bien, el usuario de la nube debería contar con la **información suficiente para dictaminar si un servicio *cloud* es aceptable o no**. Además, le sería de utilidad contar con un **conjunto de herramientas y metodologías para adecuar sus necesidades de seguridad y privacidad** al servicio suministrado por un proveedor *cloud* específico [5].

Un contexto crítico para la privacidad del usuario es la gestión de eventos de calendario mediante el *cloud*, ámbito en el que se desarrollará este Trabajo Fin de Máster. En específico, **en este TFM se diseñará e implementará una aplicación**, para ofrecer a los usuarios de la nube un sistema de **gestión de calendarios basado en la seguridad**, y que permita que los datos de los eventos permanezcan **confidenciales incluso para el proveedor de servicios**. Con todo lo expuesto, se pretende poner de manifiesto que la seguridad y privacidad son tareas que conciernen al usuario y a todo elemento que esté conectado a la red global, y que es posible adoptar medidas para **combinar de modo adecuado utilidad, seguridad y privacidad**.

1.2. Objetivos

El objetivo principal de este trabajo es intentar demostrar que se **puede desarrollar una aplicación que tenga en cuenta tanto la seguridad como la privacidad como parte del diseño integral de la misma**. Esta finalidad se persigue en la creación dentro de un entorno concreto, como es la **gestión de eventos de calendario en la nube**. La aplicación permite gestionar los eventos personales del calendario a través de una interfaz coherente y usable, de forma que los usuarios con los mecanismos adecuados puedan proteger la seguridad y privacidad de su agenda personal de modo eficaz. Además, se busca la sencillez a la hora de diseñar e implementar las funcionalidades, para que un gran número de personas pueda disfrutar de las capacidades que otorga el sistema, sin la necesidad de tener grandes conocimientos relativos a informática, criptografía o seguridad. El proceso de protección de la privacidad de la información por parte del usuario sigue un mecanismo simple y transparente, minimizando lo máximo posible las tareas que se deban ejecutar con su intervención, y así **permitirle gestionar sus eventos de manera fácil y clara en el lado del cliente**. No obstante, este proceso es más complejo internamente, estructurando y cifrando toda la información personal, para garantizar que su almacenamiento en un servidor externo no se convierta en un problema.

Asimismo, para llevar a cabo la implementación **se analizaron aplicaciones similares en el mercado**, con la finalidad de ofrecer a los usuarios un sistema nuevo, pero que no reduzca las funcionalidades que puedan estar presentes en estos otros servicios. De la misma manera, se investigaron **lenguajes de programación, entornos de aplicación, estructuras para el almacenamiento de los datos y procesos de seguridad**. La herramienta desarrollada ha seguido una **filosofía *OpenSource***, como vía para la **reducción de costes de desarrollo**, pero también para favorecer su **evaluación y mejora colaborativa** [6].

La conjunción de estos elementos pretende incrementar los conocimientos y la experiencia personal obtenida hasta el momento acerca de los campos de estudio del trabajo y la informática. En este respecto se ha tenido muy en cuenta que la seguridad en la actualidad es una parte fundamental del diseño de cualquier proyecto *software* [7], y no un mero añadido que se puede incluir en las aplicaciones al finalizar la implementación. Por último, este proyecto ha exigido desarrollar una aplicación completa, incluyendo todas las fases del ciclo de vida *software*. Esto es

importante para comprender como aplicar todo el conocimiento previamente adquirido, pero también para saber gestionar los recursos existentes de cara a implementar de modo eficiente un sistema final de acuerdo con los requisitos de partida.

1.3. Planificación

A continuación, se muestra la planificación con la cual se ejecutó el trabajo. El proyecto se realizó durante cuatro meses, en los cuáles se llevaron a cabo diferentes fases para su desarrollo. Las etapas específicas que comprenden el mismo son las siguientes:

- Documentación y estudio del estado del arte

En esta primera fase se investigaron diversas referencias y medios con la finalidad de poder definir un ámbito de trabajo adecuado para la concreción del proyecto. Se establecieron varios temas de estudio sobre los cuales se profundizó en su conocimiento. La información e ideas obtenidas acerca de esta investigación fueron plasmadas en el documento.

- Análisis

Tras finalizar con la tarea anterior, se especificó el problema a abordar y una solución plausible. Con ello, se definieron los requisitos, tanto funcionales como no funcionales, de la aplicación. Asimismo, se incluyó un esquema con los casos de uso. Después, se estudiaron las herramientas idóneas para crear la aplicación de escritorio, así como la manera de desarrollar un sistema compatible con los entornos *cloud*. Para ello, se investigaron diferentes *frameworks* que cumplieran con las necesidades requeridas. Además, se analizaron las formas de crear contextos de almacenamiento local en la herramienta *software*. También se efectuó una búsqueda de nuevos mecanismos de seguridad.

- Diseño

Una vez se estudiaron las herramientas y teorías base del proyecto, y después de seleccionar los procesos y mecanismos oportunos, se procedió con la especificación del diseño. A continuación, se pasó a diseñar el cliente de escritorio, teniendo en cuenta la modularidad como criterio esencial para organizar su estructura. Con ello se busca ofrecer una vía apropiada para el mantenimiento y

posterior modificación del *software*. Del mismo modo, se definieron las estructuras de datos a manejar. También, se incluyó un diagrama de flujo de la navegación.

- Programación

Durante esta etapa se realizó la implementación del diseño concretado en la fase anterior. Para ello se emplearon las herramientas y los mecanismos seleccionados después de cumplir con su evaluación. Las necesidades de recursos no fueron altas, pues se intentó reducir lo máximo posible este aspecto. Igualmente, se generó la documentación del sistema.

- Pruebas y experimentación

En esta fase se especificó un plan de pruebas para validar el producto *software* creado. Esto permite verificar que la aplicación cumple con las características establecidas inicialmente. Además, otro objetivo es encontrar la mayor cantidad posible de errores, logrando así depurar el sistema.

- Informe y presentación

En este último estadio, se acometió la redacción de la memoria final del trabajo con la exposición de los resultados obtenidos y el *software* desarrollado. Asimismo, se elaboró la correspondiente presentación para mostrar el proceso de desarrollo del nuevo entorno.

1.4. Estructura

Esta memoria está redactada siguiendo una organización lineal, en la cual se van especificando y describiendo las etapas por las que se ha pasado durante la elaboración de la aplicación *software*. Dichas partes se dividen en capítulos y la distribución de los mismos es la que se expone a continuación:

- Capítulo 1: En este capítulo se presentan las causas y los objetivos por los que se realizó el presente proyecto. Asimismo, se describe el plan de trabajo que se ha seguido y la estructura del documento final del TFM.

- Capítulo 2: En este capítulo se expone el estado del arte, en el cuál se incluye el concepto de las tecnologías de la información y la computación en la nube. De igual forma, se presenta la evolución de estos temas y se explica su relación con la sociedad. También, se

especifican diversos aspectos vinculados con la seguridad en las tecnologías y en algunos entornos utilizados en los servicios *cloud*.

- Capítulo 3: En este capítulo se analizan los desafíos que se quieren afrontar durante la elaboración de este proyecto. Del mismo modo, se detalla la finalidad y la funcionalidad de la aplicación a desarrollar. Igualmente, se especifican las herramientas y tecnologías que se utilizarán en la implementación del sistema.

- Capítulo 4: En este capítulo se concreta el diseño a seguir para la creación de la aplicación. También, se define la secuencia entre los diferentes estados de la aplicación y las estructuras que se deben emplear para su correcto funcionamiento.

- Capítulo 5: En este capítulo se explica el proceso de la implementación del sistema, exponiendo el entorno de desarrollo utilizado, el código generado y el criterio que se ha aplicado para la documentación de la aplicación.

- Capítulo 6: En este capítulo se presenta la elaboración del plan de pruebas a seguir para la validación del *software*. Además, se exponen los resultados obtenidos de la ejecución de dicha tarea para generar unas conclusiones.

- Capítulo 7: En este capítulo se describe el resultado final de la aplicación desarrollada, mostrando su interfaz y secuencias mediante el empleo de imágenes e información específica para cada proceso.

- Capítulo 8: En este capítulo se exponen las conclusiones finales que se han obtenido y los aspectos relevantes hallados, después de dar por acabado el proyecto. Igualmente, se manifiestan diversas ideas que se podrían abordar en un trabajo futuro.

2. Estado del arte

En este apartado se va a exponer el estado del arte del trabajo, el cual requiere realizar unas tareas previas de investigación con la finalidad de precisar y contextualizar la situación de los temas a tratar en un marco adecuado. Esta labor se realiza para ofrecer una base temporal inicial, que permita continuar con la ampliación de los diferentes elementos y aspectos del proyecto. Los contenidos tratados posteriormente son una introducción de la evolución y la relación de las TI con el *cloud computing* (computación en la nube). También, se profundiza en el vínculo de estos ámbitos con la sociedad y la seguridad en la actualidad. Por último, se plasma el tema de la gestión de los datos por parte de las empresas en torno a la privacidad.

2.1. Tecnologías de la información y *cloud computing*

Las Tecnologías de la Información (TI), son un concepto muy amplio pero se podrían definir como "*el conjunto de elementos que guardan relación con los sistemas informáticos, dónde se incluye el hardware, el software, los periféricos y las redes, con la finalidad de almacenar, estudiar, proteger, recuperar, transmitir y procesar datos de manera electrónica*" [8]. Esta definición intenta abarcar todos los aspectos relevantes del tema, pero al ser un concepto que aúna diversos elementos, siempre habrá diferentes interpretaciones. Además, al ser un ámbito en continua evolución, y cada vez más rápida, las definiciones podrían presentar cambios. La primera vez que apareció este concepto fue en 1958 en una publicación hecha por los autores Harold J. Leavitt y Thomas L. Whisler en la *Harvard Business Review* [9], y es entonces donde se podría establecer el punto de partida para la explicación del término y su significado moderno.

Los inicios de las TI son un poco inciertos, pero ya desde la antigüedad las diferentes culturas han realizado distintos procesos, de una u otra manera y más o menos complejos, con el propósito de comunicarse y perpetuar la información [10]. Durante todos estos años la evolución de los recursos relacionados con estas tecnologías ha sido constante y de forma gradual, pero desde mediados del

siglo XX estos elementos han tomado un cambio vertiginoso, con el objetivo de crear sistemas cada vez más sencillos y con unas capacidades de procesamiento, almacenamiento y comunicación más notables. Los inconvenientes destacados a solventar eran el tamaño de las máquinas, la volatilidad de la memoria, y la capacidad y velocidad de cálculo.

Dentro del anterior contexto, en las últimas décadas, aparece el término de computación en la nube (*cloud computing)*, el cual guarda una estrecha relación con las TI [11]. Estos entornos se concibieron como una gran estructura conectada a la red, donde sus dispositivos se encargaban de realizar las tareas de procesamiento y cálculo, para así suministrar a los usuarios servicios libres de las cargas de administración y mantenimiento [12]. Sus orígenes fueron relacionados con los investigadores John McCarthy y J.C.R. Licklider, los cuales tenían ciertas ideas acerca de cómo sería el concepto de la computación en la nube e Internet [13]–[15]. El progreso de las tecnologías hizo llegar uno de los primeros hitos importantes para el desarrollo de estos sistemas. En 1969 surgió Internet [16], desarrollada por el Departamento de Defensa de los Estados Unidos. En esa época la red se creó para comunicar algunos organismos de esta nación. La base del proyecto era montar una red descentralizada con varias rutas, donde la información enviada entre dos lugares, fuera segmentada y transmitida por diferentes caminos hasta alcanzar su destino. Después se fueron añadiendo más funcionalidades a la red, y se incluyeron servicios de mensajería y páginas web. A finales del siglo XX el invento de Internet provocó un perfeccionamiento de las redes de intercambio de información, que empezaron a ser accesibles para todo el mundo, dando lugar a nuevas posibilidades para las personas (Ver Figura 1).

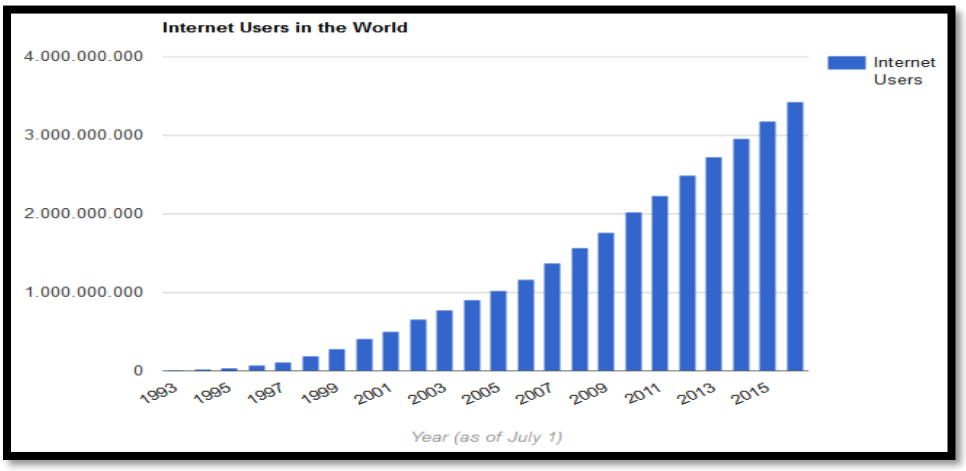

Figura 1: Evolución usuarios globales Internet
Fuente: http://www.internetlivestats.com/internet-users/#trend

A pesar del avance conseguido hasta el momento, se observó que en el camino de la mejora de estos entornos era necesario un paso más en la innovación de la tecnología, para así crear los servicios *cloud* de un modo adecuado [17]. Pero esto no ocurrió hasta unos años más tarde y ya entrado el siglo XXI, cuando los equipos (*hardware*, en general) y las conexiones (Internet de banda ancha y las redes móviles) mejoraron y se optimizaron [18]. Es en este momento cuando de verdad empieza el despegue del *cloud computing*.

Las primeras empresas que se atrevieron a acoger estos entornos fueron Amazon, Microsoft, Salesforce y Google. Esta última, con su servicio *Google Docs* anunciado en 2007, hizo que la mayoría de la gente conociera la existencia de estas nuevas aplicaciones. A partir de ahí se intensificó el desarrollo de estos entornos y se empezaron a establecer diferentes capas sobre el ecosistema, dando como resultado la capa de aplicaciones y *software* (*SaaS*), la capa de plataformas (*PaaS*) y la capa de infraestructuras (*IaaS*). Dentro de estas capas se desplegaron todo tipo de servicios como, por ejemplo, servicios de comunicación, de almacenamiento de datos, de alquiler de entornos virtuales y *hardware*, de distribución de contenidos, y de gestión de eventos de calendario (Ver Figura 2). Esto fue debido gracias a la gran diversidad de elementos vinculados con la informática que se pueden ofrecer a través de servicios *cloud* [19].

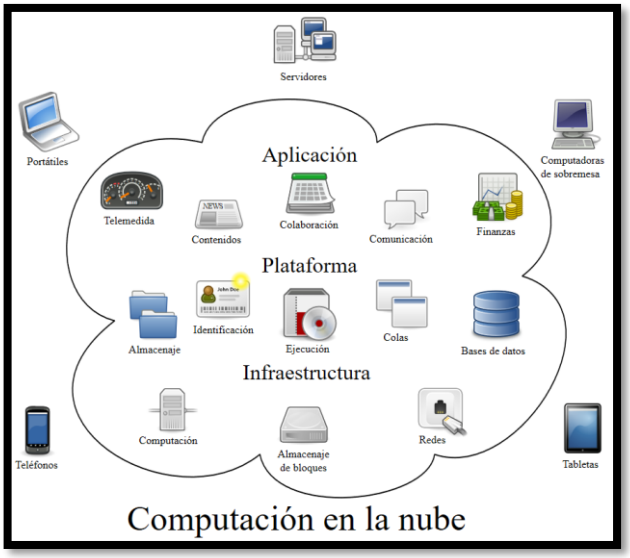

Figura 2: Ilustración de servicios *cloud*
Fuente: https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube

Siguiendo con la expansión de las tecnologías *cloud*, Google aprovechó para crear un sistema operativo orientado a funcionar siempre conectado a Internet y cuyas aplicaciones fueran servicios en la nube, este sistema es *Chrome OS* [20]. Si bien es cierto que este entorno al principio no tuvo una buena acogida, con el paso de los años ha visto cómo su popularidad ha ido creciendo hasta hacerse un hueco en el mercado actual, incluso en algunas regiones y ámbitos es un sistema operativo muy utilizado [21]. Además, este tipo de enfoque para los sistemas y las aplicaciones se ha ido extendiendo, dando lugar al inicio de un nuevo paradigma de desarrollo y funcionamiento.

Por último, la creciente facilidad de acceso a Internet ha originado una rápida evolución de las tecnologías y su gran aceptación en muchos entornos, en los cuales la interacción entre las personas involucradas ha derivado en la configuración de nuevo espacio social, la *sociedad de la información* [22].

2.2. Entornos *cloud* y la sociedad

Hoy en día, los servicios suministrados por el *cloud computing* han entrado fuertemente en las vidas de las personas, haciendo que todo tipo de gente, sin importar el género ni la edad, haga uso de sus características y ventajas [23]. Estos sistemas les permiten realizar sus trabajos cotidianos con mayor facilidad, lo que provoca por un lado una gran adopción de las aplicaciones, pero por otro se crea una dependencia real con su uso. Consecuentemente, los entornos *cloud* han modificado de manera importante el sistema social, cultural y económico, haciendo que las personas cambien su perspectiva para afrontar su realidad, tanto personal como laboral [24].

Desde que empezara la evolución de estas tecnologías, el progreso en los servicios ha sido constante, pero lo más sorprendente son los vínculos que se han creado entre las personas y su información (Ver Figura 3). Correspondientemente, las aplicaciones se han tenido que ir adaptando a esta nueva forma de coexistencia para seguir siendo útiles para los usuarios.

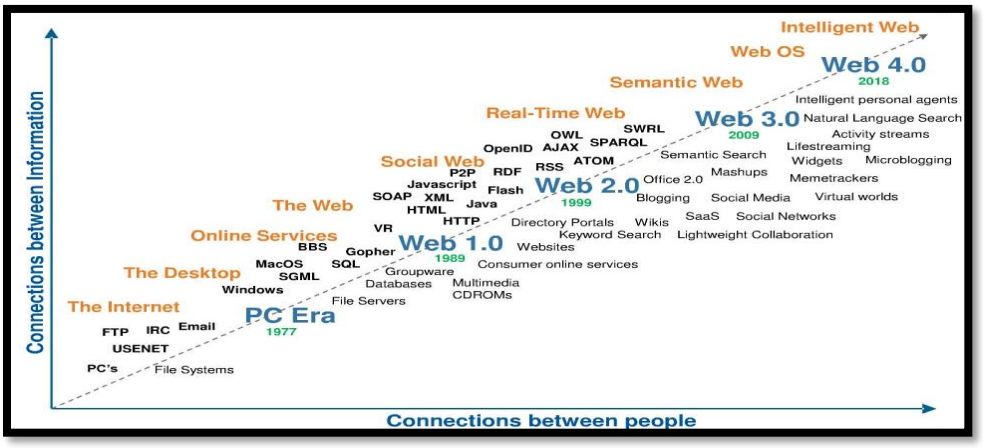

Figura 3: Ilustración entre las conexiones de personas e información
Fuente: https://www.prokarma.com/blog/2014/10/16/what-exactly-web-30

De tal forma, las empresas que ofrecen los servicios han realizado los cambios oportunos en las funcionalidades y características de los sistemas para aumentar las posibilidades de uso y facilitar que los usuarios tengan una gran diversidad de servicios.

Además, el *cloud computing* no es algo que sólo se emplee en un contexto particular, sino que también en el ámbito privado los sistemas *cloud* son entornos que tienen presencia en distintos tipos de organizaciones de la sociedad, como pueden ser los gobiernos, los negocios y empresas, los hospitales y los centros educativos [25], [26]. Su labor se basa en suministrar a los clientes y usuarios, aplicaciones que sean capaces de trabajar en los equipos *hardware* finales para facilitar y realizar sus tareas. Asimismo, el uso de estas tecnologías busca solventar algunos problemas relacionados con la instalación de las aplicaciones, la infraestructura necesaria para el funcionamiento y, los costes y el mantenimiento derivados de los recursos. Por ello, se intentan emplear servicios *cloud* externos para que sean otras empresas las que se encarguen de las tareas principales, como la administración de las conexiones de red y la creación de las aplicaciones *software*. Igualmente, éstas mismas se encargan de ofrecer servicios fiables, seguros, eficaces y usables para que las organizaciones utilicen sus sistemas en lugar de otros, y con las garantías de que los inconvenientes son los mínimos posibles. Esta visión de trabajo hace que los organismos se puedan centrar en incrementar sus oportunidades de desarrollo comercial y públicas.

En el marco educativo gracias a las tecnologías en la nube se han conseguido nuevas formas de impartir clases ofreciendo un gran beneficio para multitud de personas [27]. Pero relacionado con el incremento de estos nuevos entornos ha venido una alta demanda de personal cada vez más cualificado que sea capaz de ayudar con el desarrollo de estos sistemas *cloud*. Se han creado nuevas profesiones y puestos de trabajo a desempeñar relacionados con este contexto [28]. A consecuencia de este panorama, en los centros de enseñanza se han empezado a impartir clases, cada vez a edades más tempranas, sobre programación e informática [29]. Pero no es hasta edades posteriores, donde a través de una titulación o capacitación sobre estas tecnologías, cuando se pueden conseguir las habilidades y conocimientos suficientes para poder gestionar de modo completo estas aplicaciones en sus diferentes aspectos. Por ello se han buscado nuevas formas de acceso a la enseñanza y que los estudiantes puedan elegir como se forman [30]. Incluso fuera del ámbito académico se está fomentando el conocimiento de las tecnologías informáticas [31]. Si bien es cierto que todo esto

se hace para formar a las personas para un futuro tecnológico, ésta no es la única finalidad, ya que también se hace para concienciar a la gente acerca de las herramientas informáticas que manejan.

Al fin y al cabo, los entornos en la nube se han vuelto populares por una serie de ventajas [32]. Estos sistemas permiten establecer nuevos modos de trabajar apoyados en la flexibilidad, la agilidad, y la adaptación a distintas necesidades y tamaños. Los recursos pueden ser obtenidos bajo demanda y se paga por lo que se consume, lo que ayuda en la reducción de los costes. Además, siguiendo con esta vía, al emplear el *cloud computing* se reduce la necesidad de caros equipos propios y de preocuparse de instalaciones y mantenimientos. En este nuevo paradigma, los servicios, las aplicaciones y los datos están disponibles de manera descentralizada, dando a los usuarios la posibilidad de acceder a los recursos en cualquier lugar y equipo, facilitando así la movilidad y no depender para la utilización de los sistemas estar en un recinto específico. También, las empresas suministran servicios globales y con una alta disponibilidad, para que los usuarios puedan utilizar las aplicaciones de forma rápida y sin restricciones, evitando la necesidad de conocer cómo se despliegan los servicios o tener que poseer grandes conocimientos acerca del tema para utilizar los entornos. Igualmente, los proveedores de los servicios al tener centralizadas las aplicaciones en unos cuantos lugares, son capaces de obtener una reducción en los tiempos de actualización y pueden buscar la mejora de la seguridad de un modo más sencillo.

Pero no todo son ventajas, también existen inconvenientes [32]. Como la necesidad imperativa de estar siempre conectados a Internet o la dependencia de los proveedores de los servicios, ya que estos suministran las aplicaciones y almacenan los datos. Además, la curva de aprendizaje de los servicios se tiene que modificar cada cierto tiempo, debido a que se suelen modificar las interfaces con el objetivo de mejorar la adaptabilidad y funcionalidad. También, la seguridad de la información puede verse afectada, pues las conexiones no se realizan únicamente entre dos puntos, sino que se recorren más elementos hasta llegar al destino. Esto provoca que en algunas ocasiones se encuentren vulnerabilidades que pueden ser explotadas, y se pueda obtener la información de los usuarios [33].

Finalmente, el uso de estas nuevas herramientas ha generado que se creen muchos temas de debate alrededor de ellas, como puede ser la obtención de datos por parte de terceros, la monitorización de los hábitos e información de las personas, la libertad de los usuarios, la dependencia de los proveedores, o la generación de entornos maliciosos. La mayoría de las cuestiones surgidas, casi

siempre, están relacionadas con la seguridad o la privacidad, por ello hay que poner énfasis en solventar estos asuntos y asumir mayor transparencia, para evitar que la sociedad vea con desconfianza estas tecnologías. Pero esto no es un problema de la época moderna, la iniciación de que estos aspectos sean causa de discusiones éticas y morales fue en la década de 1940, cuando Norbert Wiener introdujo esta problemática en las TI [34].

2.3. Entornos *cloud* y la seguridad

Tras instaurarse la democratización de la tecnología, uno de los aspectos de los entornos *cloud* que realmente ha cobrado mayor importancia en los últimos años es la seguridad, y esto es debido a la ocurrencia de varios incidentes con bastante relevancia y a su mediatización [35], [36]. Internet no es un sitio confiable en un principio, y aún con ello había algunas empresas que no veían la seguridad como un tema prioritario. Hace muy poco tiempo algunas aplicaciones no aseguraban la información de sus usuarios. Las conexiones y el intercambio de los datos se realizaban de manera legible para cualquier persona en la red [37]. Además, muchas empresas no tienen en cuenta la seguridad en el diseño de sus servicios y esa es la causa por la que las aplicaciones sufren una gran exposición a los ataques malignos [38].

La seguridad en los servicios *cloud* se persigue para intentar minimizar y eliminar las amenazas y los riesgos, logrando así evitar que terceras partes logren conseguir un acceso indebido a los recursos donde se despliegan las aplicaciones [39]. La finalidad de las medidas que se adopten es impedir que los agentes maliciosos obtengan la información de los usuarios, arrebatándoles la propiedad, e imposibilitar que dichos datos puedan ser empleados para llevar a cabo actividades inadecuadas [40]. No obstante, siempre puede haber fallos y los atacantes lograrán alcanzar sus objetivos [41], [42], lo que demuestra que cualquier medida en seguridad nunca es suficiente. Esto ha provocado que algunas empresas se tomen este tema con seriedad y lleguen a ofrecer recompensas, en algunos casos bastante elevadas, para que se les ayude con la localización de errores en sus servicios [43].

La búsqueda de vulnerabilidades en este tipo de sistemas y los ciberataques se han intensificado en la última década (Ver Figura 4), pero se tienen indicios de que la vulneración de los sistemas

de información se lleva efectuando desde su origen, si bien es cierto que el destino y el fin de tales ataques ha cambiado con los años, y que ahora el proceso se ha automatizado [44]. Parte del crecimiento de este problema ha sido el auge de Internet y el fácil acceso a las redes. Por ello, estos entornos en la nube tienen que ser supervisados constantemente por las empresas propietarias de los servicios, pues al volverse un objetivo principal, las aplicaciones que suministran a los usuarios pueden ver reducida su disponibilidad por las dificultades que presenten los atacantes [45]. Los dispositivos de control y protección contra estas adversidades son muy variados, pero algunos de ellos son: los antivirus, los *IPS*/*IDS*, los *firewalls*, los *proxys* y los balanceadores de carga.

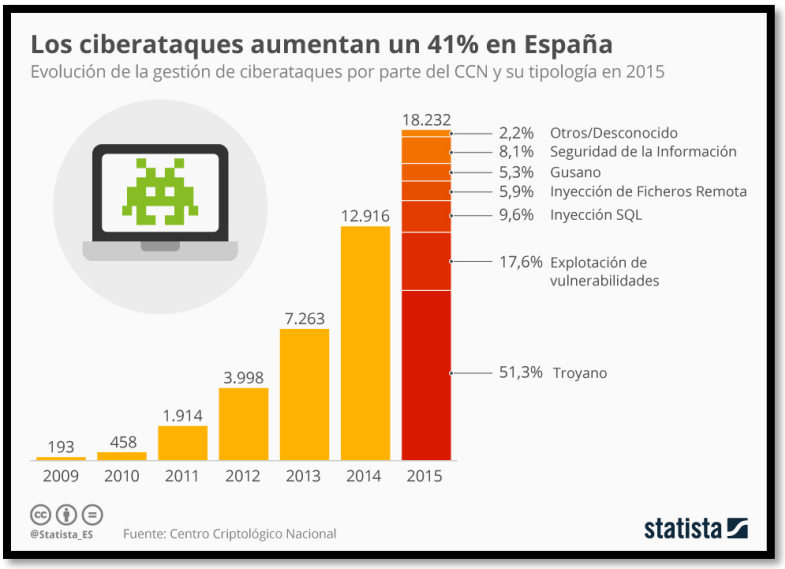

Figura 4: Evolución ciberataques en una región
Fuente: https://es.statista.com/grafico/6876/los-ciberataques-en-espana/

Los ciberataques consisten en explotar alguna vulnerabilidad de los equipos para lograr acceder a los mismos y así introducir o producir una amenaza en el sistema. Algunos ejemplos de los tipos más comunes son: los virus, los gusanos, el *phishing*, el *spam*, los ataques *DDoS*, la inyección de código *SQL*, el *XSS* y el *ransomware* [46]. Con el incremento de la diversidad de dispositivos y entornos conectados a Internet, como pueden ser coches, métodos de pago electrónicos, dispositivos *wearables*, y electrodomésticos, la protección es una tarea cada vez más difícil [47].

Además, es frecuente pensar que estos hechos maliciosos son perpetrados por uno o varios individuos localizados, pero esto está cambiando, ya que los ataques se están viendo cometidos por grupos organizados de todo tipo, *hacktivistas*, criminales, terroristas e incluso países [48].

Teniendo como objetivo cualquier elemento que pueda producir un daño o un impacto importante en las personas [49], [50], alterando el equilibrio de poder y obteniendo cierta ventaja estratégica. Esto ha dado paso a que el nuevo campo de batalla sea la red y se conozca a este fenómeno como una *Ciberguerra* [51].

La creciente complejidad de la red y la dificultad para identificar los agentes involucrados en los flujos de información hacen que los temas de seguridad sean cada vez más importantes y vitales a la hora de suministrar buenos servicios a los clientes. Por tanto, también se ha hecho necesario la capacidad de ofrecer a los usuarios métodos alternativos de las aplicaciones principales, dando la posibilidad, por ejemplo, de usar las herramientas de forma *offline*. Este manejo sin conexión es una buena manera de mejorar la experiencia de uso cuando la aplicación vea interrumpido su funcionamiento con los servidores. Igualmente, ofrecer dentro de los sistemas métodos de *backup* es un buen mecanismo para que los usuarios puedan obtener sus datos de modo sencillo, y así poder hacer frente a la posibilidad de secuestro de la información por parte de terceros. La seguridad tiene que formar parte de las aplicaciones sin que ésta haga que se vea afectado el servicio que se suministra. Para ello, se deben formar profesionales que sean capaces de desarrollar entornos que tengan en cuenta este aspecto desde el principio [52].

La seguridad total no existe, siempre habrá una forma de sortear las medidas que se ponen para evitar las intrusiones, pues los ataques y procesos maliciosos con el paso del tiempo son cada vez más sofisticados y precisos [53]. Las fortalezas que se crean alrededor de los datos y los sistemas son una buena manera de protección, puesto que intentan reducir los riesgos, pero a menudo estas herramientas son ineficaces, ya que toda cadena se rompe por el eslabón más débil, y a menudo éste suele ser el usuario. Debido a ello, hay que educar a la sociedad en unos hábitos de prevención y atención con los elementos relacionados con Internet. Hay que concienciar a los usuarios que los temas de seguridad y privacidad no son banales [54], [55], pues conociendo y teniendo en cuenta los principales riesgos, se disminuye enormemente la posibilidad de ser vulnerables.

Esta constante incertidumbre hace que los proveedores de los servicios se vean sometidos a un dilema, pues sus sistemas deben estar protegidos, pero el servicio que se presta a los usuarios nunca debe dejar de funcionar. En este ámbito, se busca una seguridad completa, no obstante cuanto más se aseguran los entornos, mayor es la probabilidad de ocurrencia de fallos y de que el servicio se vea interrumpido [56].

2.4. Aplicaciones *cloud* y gestión de datos

Con el comienzo de la denominada *era de la hiperconexión* y con la utilización cada vez más habitual de los servicios *cloud*, los usuarios son capaces de depositar mucha información en Internet, en un principio sin valor, pero en otras ocasiones esto no es así [57]. Esto provoca que todos los datos que se almacenan en la red sean administrados por terceros, lo que en muchos casos supone delegar cualquier responsabilidad en la custodia de tales datos sobre gente anónima. Por ello, algunas personas quieren tener mayor conocimiento acerca de cómo los proveedores de los servicios realizan la gestión de los datos. En efecto, un proveedor de servicios *cloud* protege la confidencialidad de la información cifrándola, pero mantiene bajo su control las claves criptográficas utilizadas, de forma que puede vulnerar la privacidad de los datos accediendo a ellos sin contar con el consentimiento expreso del propietario. Debido a esto, algunos usuarios quieren empezar a ser una parte activa en la custodia de sus datos [58].

Las empresas propietarias de las aplicaciones *cloud* no suelen explicar de modo claro las formas y procesos que emplean para gestionar la información de los usuarios [59], debido a esto no queda otra opción que confiar en las buenas prácticas de los proveedores de los servicios. Esta manera de administración e intento de recopilación de los datos de los usuarios se está volviendo frecuente, y ha empezado a afectar a diferentes entornos de la tecnología, como pueden ser sistemas operativos [60] y dispositivos móviles, o recursos, como documentos, correos electrónicos [61] o gestores de calendario [62]. En algunos otros contextos, como la mensajería instantánea o el almacenamiento local, la concienciación sobre la importancia de la seguridad y la privacidad ha ido en aumento, hasta el punto que se han creado distintas aplicaciones o se han añadido nuevas características para cumplir con este doble cometido.

La necesidad inagotable de las empresas de conocer los gustos y hábitos de las personas viene marcada por el afán de sacar mayor provecho lucrativo, comercial y estratégico de los datos gestionados, pues al fin y al cabo, la información es poder [63]. Contra esta situación, algunas naciones han creado leyes para intentar frenar esta continua violación de la privacidad [64] (Ver Figura 5). No obstante, este hecho en algunos casos no ha cumplido con su finalidad de contener y disuadir a las empresas para que no prosigan con dichas acciones, lo que ha provocado que las autoridades competentes de los países impongan multas por seguir realizando estas actividades de

modo flagrante. Las empresas mantienen este tipo de modelo de gestión, debido a que les suele salir más rentable pagar multas que cumplir las leyes de protección de la privacidad [65].

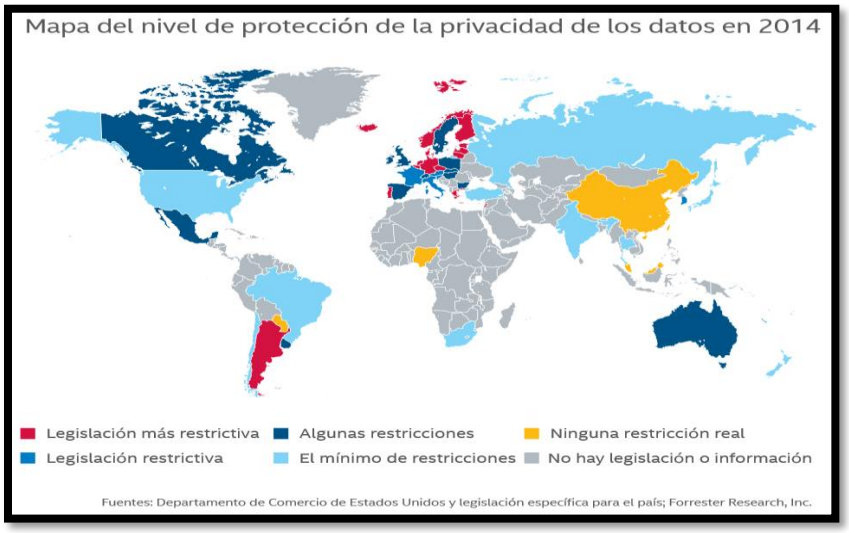

Figura 5: Ilustración de la legislación relativa a la privacidad a nivel global
Fuente: https://mcafee.app.box.com/v/2016predictions

Desde el punto de vista técnico, uno de los mecanismos oportunos para devolver la privacidad de la información a sus dueños pasaría por emplear la criptografía como herramienta para ocultar los datos a terceros [66]. En efecto, la criptografía proporciona los mecanismos básicos para la protección de la confidencialidad de la información. En términos criptográficos, dicha confidencialidad significa que solo pueden acceder a la información cifrada aquellos usuarios que tengan en su poder las claves criptográficas pertinentes. De esta forma, si se quiere proteger la privacidad de los datos frente a un proveedor *cloud*, bastaría con impedir que dicho *cloud* tuviera acceso a las claves criptográficas empleadas al cifrar la información. Aquí caben dos soluciones. Se puede optar por emplear *cifrado homomórfico* para que el proveedor *cloud* pueda hacer operaciones sobre los datos cifrados sin necesidad de descifrarlos. Ahora bien, esto implica que el proveedor ha de utilizar este tipo de cifrado, lo cual no ocurre en la mayor parte de los casos. Por ello, en términos generales la opción más factible consiste en cifrar la información de modo local en el lado del cliente, que es donde se generan los datos de los usuarios que utilizan los servicios. A consecuencia, se consigue que los datos manejados por los sistemas no sean legibles por nada ni por nadie y que únicamente el usuario propietario, con algún proceso concreto, pueda entender la información. Así los proveedores de los servicios no tienen manera de conocer los datos que se insertan o se almacenan en sus servidores propios. Además, este proceso también es efectivo contra

la obtención de la información a través de ataques maliciosos, ya que, si los atacantes consiguen entrar en los servidores, no podrán saber el contenido de los datos sustraídos [67]. Con todo esto se logra que las aplicaciones *cloud* sean únicamente empleadas como servicios de administración, transporte y almacenamiento de los recursos personales de los usuarios. Cambiando el enfoque de cómo se realiza la preservación de la confidencialidad de la información, empleando un nuevo modelo, donde la privacidad está sujeta a la desconfianza total de los elementos externos, tanto servidores de procesamiento y almacenamiento, como de agentes humanos, con o sin autorización [68]. Igualmente, para mejorar la percepción de transparencia del servicio de cara al usuario se pueden crear aplicaciones *OpenSource*, donde se de la libertad suficiente para conocer los detalles necesarios del sistema [69].

La gestión de los datos personales a través de la seguridad no tiene por qué estar reñida con la usabilidad que se le entrega al usuario. Así, los mecanismos de privacidad que se empleen deben ser lo suficientemente sencillos para poder abarcar los conocimientos de todo tipo de personas, logrando con ello la popularidad del servicio y la búsqueda del éxito. En esta clase de aplicaciones la intervención del usuario tiene que ser mínima y adecuada para dar la impresión de uso fácil y que se cumple con la funcionalidad requerida [70]. Esta tarea queda a cargo de los desarrolladores de los sistemas. Además, la robustez y calidad del servicio se alcanza con el uso de algoritmos y procesos idóneos en un segundo plano, garantizando que se otorga la privacidad prometida y que no se incomoda a los usuarios en su experiencia con la aplicación incluyendo una complejidad innecesaria. Ya existen herramientas que son respetuosas y buscan la privacidad de la información del usuario [71], pero éstas son una minoría en comparación con el total de aplicaciones disponibles. Una causa de este panorama es la rentabilidad, pues las empresas no están dispuestas a perder parte de sus beneficios a menos que exista una demanda social y/o una exigencia legal.

Finalmente, hay que recordar que los servicios *cloud* les pertenecen a las empresas que los desarrollan y publican para el consumo de los usuarios. Esta realidad no se puede olvidar, pues a menudo las organizaciones realizan cambios a favor de sus intereses en las condiciones de uso de las aplicaciones, y no queda otra opción que aceptar las obligaciones impuestas en dichos contratos, si se quiere seguir utilizando el sistema [72], [73]. No es lo habitual, pero a veces estas modificaciones son el producto de problemas internos, que han sido causados después de sufrir ataques maliciosos y que han puesto en peligro los datos del servicio en la nube [74].

3. Análisis

Después de observar la implicación de las TI en los entornos *cloud* y enseñar la situación de estos contextos en relación a la sociedad, la seguridad y la gestión de la información, ahora se va a tratar un dominio concreto, los servicios de calendario. De modo más específico, se aborda la necesidad de mejorar el problema encontrado en los gestores de eventos de calendario en la nube y una posible solución para esta situación. Por ello, se exponen con detalle los inconvenientes encontrados y se describen las características que se pretenden cumplir en la definición del proyecto. El concepto de la aplicación es ser un gestor de eventos de calendario, que tiene los recursos almacenados de manera local o remota, y que realiza el cifrado de la información en el lado del cliente antes de guardar los datos con la intención de asegurar la privacidad. Igualmente, parte de la solución pasa por crear un sistema *OpenSource* y con un carácter multiplataforma. Para alcanzar el propósito descrito se especificó el catálogo de requisitos, tanto funcionales como no funcionales, que debe satisfacer la aplicación a desarrollar. Además, se ilustró con un diagrama los casos de uso presentes en el sistema.

Con la finalidad de cumplir las necesidades requeridas y después de haber concretado la especificación y funcionalidad del proyecto, se realizó un análisis de diversas tecnologías y herramientas de programación que permitieran abordar los objetivos marcados. Igualmente, se investigaron diferentes formas de almacenamiento para poder ofrecer mayor diversidad en la utilización de la aplicación. Asimismo, se examinaron procesos alternativos de seguridad y de representación de la información. Incluso, se estudiaron estructuras de datos y de ficheros con la intención de administrar los datos de forma adecuada.

3.1. Descripción del problema

Para conocer las desventajas y dificultades que presentan algunos de los diferentes sistemas de gestión de eventos de calendario en la nube, se hizo un análisis de la situación y características de este tipo de aplicaciones presentes en el mercado. Los principales entornos examinados fueron

Google Calendar [75], *Outlook Calendar* de Microsoft [76] y *Calendar* de Apple [77]. Los inconvenientes que se encontraron en estos servicios fueron los siguientes:

- Falta de funcionalidad sin conexión a Internet, tanto si se interrumpe la conexión, como de la posibilidad de emplear la aplicación sin uso de datos.
- Poca información detallada y precisa en referencia a la gestión de la información, lo que impide saber si se está respetando la privacidad de los usuarios. Los temas más delicados en términos de la privacidad de los usuarios son el almacenamiento, la recopilación y la difusión de los datos.
- No ofrecen modos de personalización temática, o si lo hacen no son de manera nativa.
- No hay posibilidad de realizar *backups* manuales.
- No permiten usar aplicaciones libres y de terceros que puedan interactuar con los servicios de calendario.
- Obligación de aceptar un contrato de condiciones de uso, el cuál si no es aceptado no se puede usar el servicio suministrado. Además, este documento suele sufrir modificaciones de manera unilateral y sin previo aviso a lo largo del tiempo, favoreciendo las necesidades del proveedor del servicio.
- Estos importantes sistemas del mercado son propiedad de grandes compañías que no suelen prestar especial atención a ciertas características del usuario. Estas empresas hacen con sus aplicaciones lo que ellas quieran sin posibilidad de réplica por parte del usuario, más allá de la posibilidad de dejar de utilizar la aplicación.
- Necesidad de utilizar un dispositivo o entorno concreto para ejecutar la aplicación.

No obstante, aparte de estas aplicaciones existen otras menos conocidas. Estos sistemas comparten que son específicos para una plataforma en concreto, como puede ser *Android*, *iOS* u otro sistema operativo. Además, también hay otros métodos de emplear los servicios y es añadiendo algún *plugin* por encima para dotar al sistema de ciertas características nuevas [78]. Pero se ha observado que en el mercado no existen aplicaciones con la orientación del enfoque de este trabajo, lo más parecido que se encontró fue una aplicación *Android* denominada *Flock* [79], la cual ofrecía mantener la privacidad en los datos personales de los usuarios, como los contactos y los calendarios. Pero su desarrollo y progreso se vio un poco ralentizado, y en la actualidad parece que el proyecto

no sigue evolucionando, pues la empresa a la que pertenece se ha centrado en otros objetivos (por ejemplo, *Signal* [80]).

Por el panorama de las aplicaciones disponibles en la actualidad, se pensó en desarrollar un sistema para la gestión de los eventos de calendario en la nube con la idea de que fuese multiplataforma y *OpenSource*. Esto permitiría a los usuarios utilizar el servicio donde ellos quisieran, y a la comunidad de desarrolladores poder examinar la aplicación para verificar el funcionamiento de la misma o incluir nuevas funcionalidades. Asimismo, el sistema se centraría en ofrecer seguridad y privacidad a la información de los usuarios empleando los mecanismos adecuados y con el objetivo de fácil interacción.

3.2. Información del proyecto

Con el desarrollo de este proyecto se busca crear una aplicación de escritorio multiplataforma basada en las tecnologías web, que sea capaz de gestionar de forma segura la información de un calendario almacenado en un servidor *cloud* o de forma local. Los sistemas operativos soportados para esta tarea son: *Windows* y *Linux*. Además, con el objetivo de ofrecer la aplicación a la mayor cantidad de gente posible la interfaz debe permitir gestionar los datos de los eventos con facilidad, haciendo que las acciones entre los elementos sean intuitivas y sencillas.

El proceso inicial de la herramienta es que antes de realizar la verificación del usuario hay que seleccionar el modo de uso de la aplicación, ya sea en modo Web o en modo Local. Esto permite trabajar al sistema en dos modos de conexión, con o sin Internet. Después, el usuario debe autenticarse para acceder al sistema. Para ello, es necesario que tenga una cuenta personal, la cual se puede crear si aún no la tuviera. Los datos que se insertan en la herramienta no relativos al calendario se manejan con un cifrado simétrico. Una vez el usuario introduce sus datos y la aplicación realiza la validación correspondiente, es indispensable obtener la clave privada con la que se trata toda la información del calendario depositada en la aplicación. Esta clave es utilizada por el cifrado asimétrico empleado en la aplicación, y se genera a partir de una contraseña que establece el usuario, logrando así aumentar la seguridad. Esta clave privada se le suministra al usuario para que la guarde bajo su protección y responsabilidad. La otra clave del par, la clave

pública, se almacena en la base de datos correspondiente según el modo de uso del sistema. Este mecanismo de claves es necesario para poder ingresar dentro del sistema de gestión de eventos: todos los eventos son cifrados con AES mediante una clave aleatoria que es cifrada con la clave pública de los usuarios, de modo que dicha clave aleatoria es recuperada haciendo uso de las claves privadas correspondientes. Además, si el usuario desea cambiar su clave personal, ya sea porque ha sido comprometida de algún modo o simplemente quiere modificar este elemento, el sistema puede volver a generar un nuevo par de claves asimétricas. Cabe mencionar que al emplear esta característica los datos previos almacenados en la herramienta se vuelven inaccesibles, pues al cambiar la clave, ya no se puede recuperar la información cifrada, es decir, no es posible recuperar la clave AES aleatoria con la que se cifró el calendario. Con este proceso se consigue minimizar la intervención del usuario e involucrarlo en este tema sólo al principio. Para garantizar después la seguridad y privacidad de la información, internamente se usan los mecanismos adecuados en el sistema. Así se hace que el contenido personal del usuario se almacene de modo confidencial en la base de datos, y que únicamente sea legible en la sesión que tiene abierta en la aplicación. Con ello se persigue simplificar el proceso de cifrado y que no sean necesarios conocimientos demasiados avanzados de criptografía.

Tras superar los pasos anteriores, se muestra la pantalla principal de la aplicación, donde suceden todas las acciones importantes y se puede observar el calendario con los eventos. En esta interfaz las opciones son muy variadas. En primer lugar, se permite realizar búsqueda de eventos y ver las notificaciones sobre los eventos compartidos con los asistentes. Igualmente, se pueden administrar los eventos. Así, se permite la creación, actualización, copia de información, eliminación, carga masiva (tanto por *URL* como por fichero) y descarga de eventos. Algunas de estas funciones se pueden hacer directamente interactuando con los eventos mostrados. Además, cuando se administran los eventos se pueden compartir los mismos con tantos usuarios como se quiera, siempre y cuando estén presentes en la aplicación, debido a que es indispensable tener su clave pública para efectuar la operación. Después, cada asistente con su propia clave privada podrá recuperar la clave AES de cifrado y obtener la información del evento compartido. Finalmente, es posible visualizar de manera personalizada los eventos a través tanto de diferentes vistas (mensual, semanal, diaria o de agenda) como de distintos temas y modos de interfaz de calendario. También se puede desplazar el momento mostrado por el calendario, de forma corta, en función de la vista seleccionada, o de manera rápida, año a año.

Además, en la herramienta hay otras características, como la obtención de la clave, la ocultación del menú lateral, la visualización del calendario en pantalla completa, el borrado permanente de todos los eventos y el cierre de sesión del usuario. Toda información que se quiera comunicar al usuario se hace a través de un cuadro superior emergente, un diálogo o mediante notificaciones de escritorio. Cabe destacar que cuando la aplicación es utilizada en modo Web, si la conexión es interrumpida, el sistema sigue funcionando sin problemas. Lo que tiene que tener en cuenta el usuario es que, si la conexión no se recupera, toda la información que no se sincronice con el servidor se perderá, a no ser que se efectué una descarga del calendario con las modificaciones realizadas, para después hacer una importación de la misma dentro de la aplicación. Asimismo, es importante que cuando se restablezca la comunicación con el servidor se lleve a cabo una sincronización de los eventos. Los eventos compartidos durante la fase de desconexión no se enviarán, comunicando esta situación al usuario, pero cuando todo vuelva a la normalidad los eventos serán compartidos con los asistentes correspondientes.

Por último, para favorecer la difusión dentro de la comunidad de desarrolladores y a su vez reducir lo máximo posible los costes en la implementación, la aplicación usa y se crea con una visión *OpenSource*, dando la posibilidad de auditar el sistema o modificarlo según las necesidades y habilidades de cada uno. Las herramientas, mecanismos, librerías y estructuras que se empleen seguirán este enfoque, a no ser que no haya otra opción o se busque explorar nuevas formas de desarrollo y conocimiento.

3.3. Catálogo de requisitos

A continuación, se describen y explican concretamente los requisitos que debe cumplir la aplicación de escritorio multiplataforma para la gestión de los eventos de calendario. Los elementos expuestos son características importantes y vitales para la correcta finalización del producto *software*. El catálogo de requisitos consta tanto de los requisitos funcionales como de los no funcionales. Los primeros son funcionalidades que debe suministrar la aplicación para poder llevar a cabo ciertas acciones concretas dentro de la misma. Los segundos son características que se incluyen para que el sistema sea lo más completo posible, y así intentar obtener una aplicación con mayor calidad y unas prestaciones idóneas para los usuarios.

3.3.1. Requisitos funcionales

La lista de elementos que se muestra posteriormente detalla los requisitos funcionales del sistema. Los elementos son especificados según el contexto donde prestan su funcionalidad en la aplicación. De igual forma, los requisitos son nombrados mediante la utilización de la expresión "RF" seguida del número que le corresponde a esa característica.

- **Acceso**

  - **RF1: Seleccionar modo de uso de la aplicación.**
    Elegir la forma de utilización del sistema, ya sea en modo Local como en modo Web. El modo Local implica el uso de la aplicación sin conexión a Internet y el modo Web se usa de forma contraria.

  - **RF2: Acceder a la aplicación.**
    Acceder al sistema mediante el uso de la cuenta personal del usuario registrada en el sistema. Los datos necesarios son el correo electrónico y la contraseña.

  - **RF3: Registrarse en la aplicación.**
    Permitir al usuario poder registrarse en el sistema mediante un nombre de usuario, correo electrónico y contraseña (mínimo seis caracteres). La cuenta es necesaria para poder acceder a la aplicación.

  - **RF4: Salir de la aplicación.**
    Salir de la aplicación limpiando todos los datos privados del usuario.

- **Seguridad**

  - **RF5: Mostrar credenciales de acceso.**
    Mostrar los datos del usuario que ha entrado en la aplicación en una parte visible.

  - **RF6: Crear clave privada del usuario.**
    Generar la clave secreta de los nuevos usuarios que usan el *software*. La clave requiere introducir una contraseña de seis caracteres como mínimo. La clave privada es un elemento indispensable para poder entrar en la aplicación.

- **RF7: Seleccionar archivo con la clave privada del usuario.**
  Permitir seleccionar el fichero de datos donde el usuario almacena la clave personal.

- **RF8: Introducir clave privada del usuario.**
  Registrar la clave personal del usuario para administrar los eventos del calendario.

- **RF9: Validar clave privada del usuario.**
  Verificar si la clave personal introducida es correcta para permitir el acceso del usuario en la aplicación.

- **RF10: Mostrar campos de contraseña del usuario.**
  Permitir ver la clave personal del usuario ingresada en los campos de contraseña.

- **RF11: Ocultar campos de contraseña del usuario.**
  Permitir esconder el contenido de la clave personal del usuario introducida en los campos de contraseña.

- **RF12: Validar campos.**
  Verificar el contenido de los distintos campos en función al contexto del formulario.

- **RF13: Obtener clave privada del usuario.**
  Permitir al usuario conseguir su clave privada.

- **Gestión**

- **RF14: Visualizar eventos del calendario.**
  Mostrar los eventos que se encuentran en el calendario personal del usuario.

- **RF15: Crear eventos del calendario.**
  Introducir nuevos eventos en el calendario del usuario. Los datos de los eventos son: título, fecha de inicio, fecha de fin, hora de inicio, hora de fin, tipo (diario u horario), estado (disponible u ocupado), color, lugar, *URL*, descripción y asistentes. Mostrar la información del usuario al crear el evento.

- **RF16: Actualizar eventos del calendario.**
  Modificar la información del evento seleccionado del calendario del usuario.

- **RF17: Eliminar eventos del calendario.**
  Borrar el evento seleccionado del calendario del usuario.

- **RF18: Obtener eventos del calendario.**
  Permitir copiar la información del evento del calendario seleccionado.

- **RF19: Arrastrar eventos del calendario.**
  Permitir modificar los datos del evento del calendario mediante el desplazamiento del mismo.

- **RF20: Alargar eventos del calendario.**
  Permitir modificar los datos del evento del calendario mediante la elongación del mismo.

- **RF21: Cancelar operación sobre el evento del calendario.**
  Permitir anular la operación que se realiza sobre el evento del calendario seleccionado antes de terminar y guardar la ejecución de la misma.

- **RF22: Cargar eventos de calendario externos mediante fichero.**
  Guardar eventos de calendario en la aplicación usando un fichero que contiene eventos externos. El fichero puede tener el contenido en texto claro o cifrado. Si el texto está cifrado, la información tiene que corresponder con la clave del usuario de la sesión vigente. Los eventos se insertan con un color seleccionado.

- **RF23: Cargar eventos de calendario externos mediante *URL*.**
  Guardar eventos de calendario en la aplicación usando una dirección *URL* donde se obtienen los eventos externos. Los eventos se insertan con un color seleccionado.

- **RF24: Cancelar operación de carga de eventos del calendario.**
  Permitir anular la operación de carga de eventos externos antes de finalizar y guardar la ejecución de la misma.

- **RF25: Descargar eventos del calendario en formato claro.**
  Permitir al usuario guardar los datos del calendario en un fichero en formato plano.

- **RF26: Descargar eventos del calendario en formato cifrado.**
  Permitir al usuario guardar los datos del calendario en un fichero con el texto cifrado.

- **RF27: Cancelar operación de descarga de eventos del calendario.**
  Permitir anular la operación de descarga de eventos del calendario antes de finalizar la ejecución de la misma.

- **RF28: Eliminar calendario.**
  Borrar todos los eventos del calendario personal del usuario.

- **RF29: Sincronizar calendario.**
  Sincronizar los eventos del calendario de forma manual con el servidor. Esta opción es útil después de recuperar la conexión en el modo Web.

- **RF30: Gestionar pérdida de conexión.**
  El sistema debe ser capaz de seguir funcionando a pesar de perder la conexión en el modo Web. Los datos del calendario durante esta situación son almacenados de forma local hasta poder sincronizar los mismos con el servidor. Si la conexión no se recupera, la información sin sincronizar no es almacenada en ningún sitio. Es recomendable guardar los datos de forma manual si no se vuelve a tener Internet. Los eventos descargados pueden ser introducidos en la aplicación a través del archivo obtenido.

- **Visualización**

  - **RF31: Gestionar visualización del calendario.**
    Mostar los eventos del calendario de distintas maneras. Las formas de visualizar el calendario son: mensual, semanal, diaria y agenda. Ofrecer información de la vista y marco temporal seleccionado.

  - **RF32: Mostrar información de los eventos del calendario.**
    Presentar al usuario el contenido del evento del calendario seleccionado.

  - **RF33: Gestionar la visualización del menú lateral.**
    Permitir al usuario mostrar u ocultar el contenido del menú lateral de la aplicación.

  - **RF34: Gestionar la visualización de la aplicación.**
    Permitir al usuario minimizar, maximizar y poner la aplicación en pantalla completa.

  - **RF35: Seleccionar tema de la aplicación.**
    Permitir elegir diferentes temas de personalización para la aplicación.

- **RF36: Seleccionar modo del texto.**
  Ofrecer la opción de modificar el color del texto de los eventos que se muestran en el calendario principal. Las opciones se establecen en un color claro y otro oscuro.

- **RF37: Mostrar indicador tiempo.**
  Mostrar en las vistas adecuadas un marcador temporal que indique la fecha y hora concreta de la aplicación.

- **Navegación**

- **RF38: Desplazar calendario de forma anual.**
  Mover la vista del calendario un año hacia delante o hacia atrás respecto a la representación temporal actual.

- **RF39: Desplazar calendario de forma unitaria.**
  Mover la vista del calendario una unidad hacia delante o hacia atrás respecto a la representación temporal actual. La unidad de desplazamiento varía en función a la colocación actual del calendario.

- **RF40: Desplazar calendario al día actual.**
  Mover la vista del calendario a la posición del día actual en ese instante.

- **RF41: Integración de widget de calendario.**
  Permitir realizar operaciones de navegación sobre un elemento de calendario secundario.

- **RF42: Interacción con los elementos del calendario principal.**
  Permitir al usuario seleccionar los elementos del calendario, tales como el número de la semana o el día. Esta acción realiza una modificación en la vista del calendario.

- **Búsqueda**

- **RF43: Buscar eventos en el calendario.**
  Encontrar los eventos del calendario que contengan en su título la cadena de caracteres de la búsqueda realizada.

- **RF44: Recargar búsqueda de eventos.**
  Actualizar la búsqueda de eventos realizada cuando se modifique el contenido del calendario del usuario.

- **RF45: Finalizar búsqueda de eventos.**
  Borrar la búsqueda realizada por el usuario.

- **Información**

- **RF46: Mostrar estado de la conexión.**
  Informar de forma visible el estado de la conexión a Internet, tanto si se está conectado o desconectado de la red. Este aspecto es importante en el inicio y en el modo Web de la aplicación.

- **RF47: Informar de los eventos del sistema.**
  Generar diálogos de aviso de los eventos que ocurren en la aplicación. Comunicar al usuario información relativa al sistema mediante cambios de estados en los diálogos.

- **RF48: Agrupar notificaciones de eventos compartidos.**
  Mostrar la información de los eventos compartidos del usuario de forma localizada e indicando la totalidad de eventos.

- **RF49: Notificar el estado de la compartición de eventos.**
  Mostrar la información del estado final de la compartición de eventos con otros usuarios.

- **RF50: Notificar el estado de la carga de eventos compartidos.**
  Mostrar al usuario la carga de eventos compartidos en la aplicación. Asegurar que la notificación sea visible.

- **RF51: Mostrar notificaciones de estado de eventos compartidos.**
  Visualizar las notificaciones del sistema sobre los eventos compartidos.

- **RF52: Ocultar notificaciones de estado de eventos compartidos.**
  Esconder las notificaciones del sistema sobre los eventos compartidos.

3.3.2. Requisitos no funcionales

Tras finalizar con la exposición de los requisitos funcionales, en esta sección se van a especificar los requisitos no funcionales de la aplicación. Las características de este contexto se van a detallar mediante el uso del término "RNF" acompañado del número al que corresponde el elemento. Además, se van a agrupar en función al ámbito donde se puede delimitar su tarea.

- **Ejecución**

  - **RNF1: Seguridad.**

    La aplicación tiene que gestionar los datos de los usuarios de modo correcto para que no se pueda discutir la seguridad, confidencialidad e integridad de los mismos.

  - **RNF2: Usabilidad.**

    El sistema tiene que ser intuitivo y directo en la manera de ofrecer sus características. Esto obliga a que la aplicación tenga que ser funcional y adaptable en los distintos contextos.

  - **RNF3: Interfaz.**

    Las pantallas de la aplicación tienen ser fáciles de manejar, sencillas en su aspecto y no obstaculizar la utilización del sistema con demasiada información.

  - **RNF4: Robustez.**

    El sistema tiene que hacer un adecuado manejo de los errores que se produzcan para no representar un problema en su utilización para el usuario. Además, es importante no dejar información sensible al alcance de terceros.

  - **RNF5: Rendimiento.**

    La aplicación tiene que ser lo más ligera posible para funcionar de forma rápida y fluida. La administración de los datos con el servidor ha de ser eficaz para no ocasionar pérdidas de tiempo e insatisfacción en los usuarios. Aunque esto depende de la calidad de la conexión a Internet del equipo. Los procesos en el entorno local han de aplicar las mismas características.

  - **RNF6: Interoperabilidad.**

    La aplicación tiene que poder gestionar datos sobre eventos de calendario de otros sistemas sin que esto resulte un problema.

- **RNF7: Disponibilidad.**

  La aplicación tiene que permitir al usuario usar el sistema en todo momento, ofreciendo un alto grado de acceso a los datos.

- **RNF8: Accesibilidad.**

  La aplicación tiene que poder ser usada por distintos tipos de personas. Por ello, los mecanismos y procesos del sistema tienen que ser lo más sencillos posibles. Igualmente, se debe permitir personalizar la herramienta para adecuarse a los usuarios.

- **RNF9: Estabilidad.**

  La aplicación no tiene que presentar detalles de interrupciones o cuelgues inesperados. Los elementos deben ser constantes en su apariencia y funcionalidad.

- **Evolución**

- **RNF10: Mantenibilidad.**

  El mantenimiento es un aspecto fundamental en toda herramienta *software*. Además, en este caso al tratarse de una aplicación *OpenSource*, el sistema tiene que brindar una serie de características útiles para los desarrolladores, como son la traza de información y de errores, un código bien documentado, y una estructura modular adecuada para poder incluir nuevas funcionalidades de manera sencilla. Además, como en todo producto *software* hay que buscar que el código tenga la máxima cohesión y el mínimo acoplamiento entre sus elementos.

- **RNF11: Portabilidad.**

  La aplicación es multiplataforma, y por ello se puede utilizar en diferentes entornos, tanto en sistemas operativos *Windows*, como *Linux*. Funcionando de modo correcto independientemente del sistema operativo que se tenga en el equipo.

- **RNF12: Escalabilidad.**

  La herramienta *software* tiene que estar dotada de una estructura apropiada para permitir añadir nuevas características y elementos de forma sencilla y eficaz.

- **RNF13: Coste.**

  Los costes de desarrollo necesarios no tienen que ser elevados para mantener la viabilidad del sistema. Por ello, hay que buscar el uso de herramientas libres de licencias.

3.4. Diagrama de casos de uso

En este apartado se muestra el diagrama de casos de uso (Ver Figura 6), cuya finalidad es representar el proceso de las acciones y funcionalidades que el usuario puede ejecutar cuando utilice la aplicación a desarrollar, un sistema para la gestión de eventos de calendario de forma segura. Los casos de uso que se ilustran en el diagrama son el resultado de la información especificada en los requisitos funcionales del catálogo del sistema. Después del diagrama, se detalla el comportamiento y el contexto de varios casos de uso dentro de la aplicación.

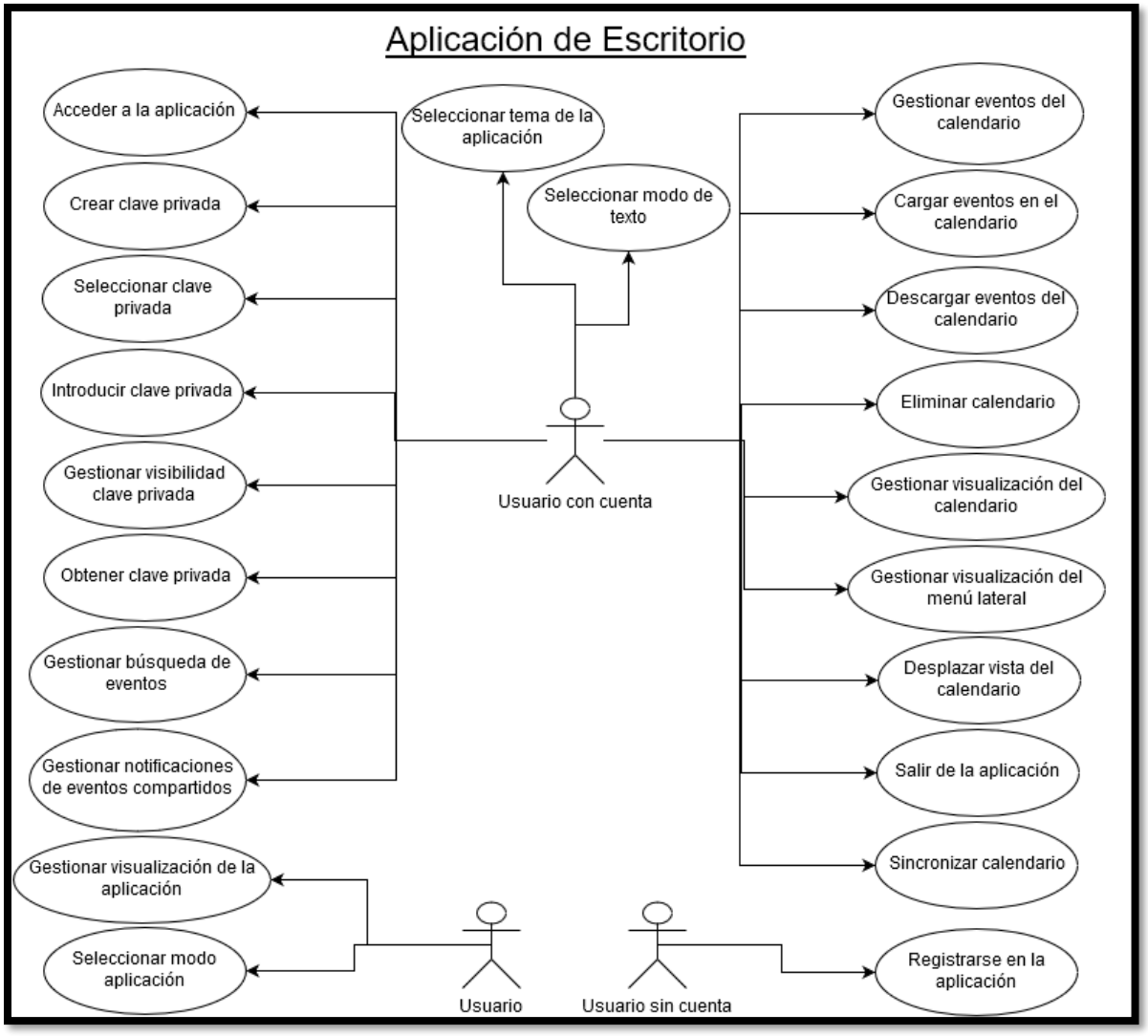

Figura 6: Diagrama de casos de uso

Seguidamente, se aclaran y explican algunos de los casos de uso más significativos dentro de la aplicación de escritorio multiplataforma.

- **Seleccionar modo de la aplicación**

  Actor: Usuario.

  Descripción: El usuario selecciona el modo de utilización de la aplicación, Web o Local.

  Precondiciones: El usuario tiene que abrir la aplicación.

  Postcondiciones: El usuario ve como la aplicación deja seleccionada la opción elegida.

- **Registrarse en la aplicación**

  Actor: Usuario sin cuenta.

  Descripción: El usuario ingresa sus datos personales y credenciales para poder registrarse en la aplicación.

  Precondiciones: El sistema no tiene una cuenta con los datos suministrados.

  Postcondiciones: El usuario es registrado y ve la pantalla para introducir su clave privada.

- **Acceder a la aplicación**

  Actor: Usuario con cuenta.

  Descripción: El usuario introduce su cuenta personal para ingresar en el sistema.

  Precondiciones: El usuario tiene una cuenta para acceder a la aplicación.

  Postcondiciones: El usuario ve la pantalla para introducir su clave privada.

- **Crear clave privada**

  Actor: Usuario con cuenta.

  Descripción: El usuario elige la opción para que la aplicación le suministre una clave privada.

  Precondiciones: El usuario ha iniciado sesión en el sistema.

  Postcondiciones: El usuario ve en la interfaz del sistema la clave privada generada.

- **Introducir clave privada**

  Actor: Usuario con cuenta.

  Descripción: El usuario registra su clave privada para poder gestionar sus eventos de calendario.

Precondiciones: El usuario ha iniciado sesión en el sistema.

Postcondiciones: El usuario ve la pantalla principal de la aplicación con su calendario.

- **Obtener clave privada**

  Actor: Usuario con cuenta.

  Descripción: El usuario selecciona la opción de conseguir su clave privada.

  Precondiciones: El usuario se encuentra en la pantalla principal de la aplicación.

  Postcondiciones: El usuario obtiene la información de su clave privada.

- **Gestionar búsqueda de eventos**

  Actor: Usuario con cuenta.

  Descripción: El usuario introduce una búsqueda a realizar entre los eventos de su calendario. Puede borrar la búsqueda después de haberla realizado.

  Precondiciones: El usuario se encuentra en la pantalla principal de la aplicación.

  Postcondiciones: El usuario visualiza los eventos encontrados con la búsqueda realizada.

- **Gestionar eventos del calendario**

  Actor: Usuario con cuenta.

  Descripción: El usuario puede ver, crear, actualizar, obtener y eliminar un evento del calendario.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.

  Postcondiciones: El usuario ve el resultado de la operación realizada sobre el evento.

- **Cargar eventos en el calendario**

  Actor: Usuario con cuenta.

  Descripción: El usuario introduce una fuente con eventos para cargar en la aplicación.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.

  Postcondiciones: El usuario visualiza el calendario con los eventos cargados.

- **Descargar eventos en el calendario**

  Actor: Usuario con cuenta.

  Descripción: El usuario elige la función para descargar sus eventos del calendario.

Precondiciones: El usuario se encuentra en la pantalla principal de la aplicación.

Postcondiciones: El usuario obtiene un fichero con los datos de su calendario.

- **Eliminar calendario**

  Actor: Usuario con cuenta.

  Descripción: El usuario elige la opción de borrar toda la información del calendario.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.

  Postcondiciones: El usuario visualiza el calendario vacío en su totalidad.

- **Sincronizar calendario**

  Actor: Usuario con cuenta.

  Descripción: El usuario elige la opción de sincronizar la información del calendario con el servidor. Esta opción está disponible únicamente en el modo Web. Es una función útil después de recuperar la conexión a Internet.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema, con el modo Web de la aplicación y la conexión a Internet se ha recuperado.

  Postcondiciones: El usuario visualiza un mensaje con el estado del proceso de sincronización.

- **Seleccionar tema de la aplicación**

  Actor: Usuario con cuenta.

  Descripción: El usuario decide cambiar el aspecto de la aplicación eligiendo un tema.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.

  Postcondiciones: El usuario visualiza la aplicación con el aspecto diferente seleccionado.

- **Salir de la aplicación**

  Actor: Usuario con cuenta.

  Descripción: El usuario decide cerrar su sesión actual del sistema.

  Precondiciones: El usuario se encuentra en la pantalla principal del sistema.

  Postcondiciones: El usuario sale de la aplicación y visualiza la pantalla de acceso.

3.5. Tecnologías y herramientas

En este apartado se muestra el análisis llevado a cabo para buscar los elementos idóneos que permitan alcanzar las funcionalidades y objetivos de la aplicación. Las principales características que se pretenden cumplir con la creación del sistema son: la seguridad, la privacidad, y un sistema multiplataforma y de código abierto. Siguiendo estas pautas, se investigaron posibles formas de gestionar la información, pues entre las inquietudes del mundo tecnológico, está la administración de los datos por parte de terceros y en entornos que no son de confianza [81]. Como consecuencia, se trataron diferentes modos de almacenamiento, estructuras de datos y formatos de fichero.

Asimismo, se estudiaron tecnologías web que apoyaran la creación de un sistema portable y flexible entre distintos entornos y equipos. Igualmente, se examinaron las herramientas adecuadas, y a poder ser, libres de licencias, para poder compartir con la comunidad de desarrolladores una aplicación *OpenSource*.

3.5.1. Aplicación de escritorio, entornos multiplataforma y tecnologías web

Las tecnologías web [82] en la actualidad son de sobra conocidas y proporcionan un gran conjunto de funcionalidades que permiten ir más allá de las tradicionales aplicaciones web, siendo utilizadas como base de aplicaciones móviles y de escritorio. Por esta razón, los lenguajes *HTML*, *CSS* y *JavaScript* han alcanzado una popularidad inesperada. Estas herramientas permiten desarrollar aplicaciones dinámicas y flexibles, pues cada una de ellas tiene un entorno de trabajo concreto y sus características han evolucionado tanto que, en sus contextos de actuación, logran grandes funcionalidades. Los ámbitos donde prestan sus servicios estos lenguajes son:

- El código *HTML* sirve para organizar y estructurar los contenidos de las páginas web. Es el esqueleto de las aplicaciones.
- El código *CSS* se utiliza para la maquetación, diseño y personalización de las interfaces de los sitios web. Dota de apariencia los sistemas.
- El código *JavaScript* sirve para gestionar y dar funcionalidad a los elementos de las aplicaciones web. Realiza las acciones de las páginas.

Debido a sus ventajas, y a que se quiere obtener mayor experiencia en el manejo de estos entornos, se decidió apostar por este enfoque. Estas tecnologías serán el núcleo principal sobre el cual se integrarán el resto de características para conseguir el *software* esperado. Las especificaciones concretas y *frameworks* seleccionados para ayudar en la creación del sistema fueron:

- *HTML5* [83], última versión estable y probada del lenguaje *HTML*. Esta versión ofrece ciertos recursos especiales, como las notificaciones, el almacenamiento de datos o algunas notaciones semánticas, que son de gran utilidad.
- *CSS3* [84], de igual forma que en el apartado anterior, es la última versión estable y probada del lenguaje *CSS*. Esta versión en concreto tiene unas particularidades idóneas que facilitar el diseño de las aplicaciones.
- En el contexto de *JavaScript*, las herramientas escogidas fueron *jQuery* [85], para el código, y *jQuery UI* [86], para añadir ciertos elementos en la interfaz.

Además, dentro de este paradigma de programación como son los entornos web, se decidió emplear un modelo de diseño de la aplicación de una sola página (*single page application, SPA*) [87], para así ofrecer una experiencia de usuario satisfactoria con unas interacciones en el sistema rápidas, dinámicas y fluidas. Este patrón lleva a cabo la carga de los todos recursos necesarios para la aplicación en una sola acción, y cuando el usuario requiere funcionalidades del sistema, éste las adquiere dinámicamente y las incluye sin mostrar interrupciones. Esta característica es fundamental para el desarrollo del *software*, ya que con ello se logra en parte dar la apariencia de que se está utilizando una aplicación de escritorio. Pero este patrón también presenta una particularidad, pues al concentrar los sistemas en menos espacio, es vital tener una buena organización para que este detalle no se convierta en un problema.

Asimismo, para completar la implementación del sistema se examinaron herramientas afines a los lenguajes web y que pudieran ser capaces de crear aplicaciones de escritorio *reales*. Estos entornos se integrarían en los diferentes sistemas operativos (*Windows*, *Linux* y *macOS*) como supuestas aplicaciones nativas, pues éstas se ejecutarían con el formato adecuado en cada uno de ellos. Con estas características concretas se encontraron dos tecnologías destacadas: *Electron* y *NW.js*.

A continuación, se ofrece una comparación de ambas herramientas (Ver Figura 7), pero los elementos a destacar de estos entornos son:

- Son tecnologías con un enfoque basado en los navegadores, concretamente, estas herramientas están relacionadas con el motor del navegador *Chrome*.

- Para ofrecer las características del lado del servidor en las aplicaciones de escritorio emplean el *framework Node.js* [88].

El *framework* de desarrollo de aplicaciones que se eligió fue *Electron* [89], pues después de realizar la búsqueda de información y comparación con otras herramientas de la misma familia, se destacó su enfoque multiplataforma y de código abierto. Aunque no menos importante es la comunidad que tiene detrás para apoyar y facilitar la resolución de posibles problemas conceptuales que puedan surgir. Otro aspecto relevante es la posibilidad de empaquetar las aplicaciones creadas, permitiendo así transportar los entornos en dispositivos de almacenamiento externos y ofrecer mayor portabilidad a los usuarios.

	NW.js 0.16.0	Electron 1.2.8
Project inception	2011	2013
Corporate Sponsor	Intel	GitHub
Licensing	Open Source, MIT License	
Browser Runtime	Chromium	libchromiumcontent
Node.js Version	6.3.0	6.1.0
Chromium Version	52.0.2743.82	51.0.2704.106
Entry Point	HTML or JavaScript[4]	JavaScript
Bare Distribution Size	139MB (52MB zipped)	125MB (45MB zipped)
Windows Platform Support	Windows 7+ (x86 and x64)	
Windows XP Support	In LTS version (0.14.x)	No
Mac Platform Support	Mac OS X.9 +	
Mac OS X.6	In LTS version (0.14.x)	No
Architecture Support	32bit (Win), 64bit (Win/Mac) & arm (limited)	
Chrome Apps Support	Yes	No
Support of chrome.* APIs	Yes	No
Plugin Support	NaCL, Pepper	Pepper
Adobe Flash Support	via Pepper Plugin	
Mac App Store Support	Yes	
Windows App Store Support	Yes	Windows 10+ (details)
App signing	Yes	
Source Code Protection	V8 Snapshot[1]	ASAR Archive Support[2]
Auto-update	Unclear (module)	Mac/Win (thru Squirrel)
Crash Reporting	No	Yes
Kiosk Mode	Partial (Buggy on Mac[3])	
PDF Viewer	Yes	Using pdf.js
Native Node Module Support	Yes	
SSL Client Certificate	Yes	Partial (details)
Print Preview	Yes	No
DevTools Extension Support	Yes	
Debugging	DevTools + extensions	Dedicated Devtron Module
Integration Testing	ChromeDriver & WebDriver	Dedicated Spectron Module
Windows Installer	Yes (nw-builder)	Yes (external module)
html5test.com Score	492	
Octane 2.0 Score[3]	27205	27343
Issue Resolution Time[6]	issue resolution 5 d	issue resolution 22 h
Open Issues[6]	open issues 29%	open issues 5%
GitHub Trends	Watch Star 31,706 Fork 3,642	Watch Star 48,846 Fork 6,863
Open Codecs/Containers	Vorbis, Theora, Opus, VP8, VP9, PCM, Ogg, WebM, WAV	
Licensed Codecs	MP3, MP4, H.264, AAC[7]	

Figura 7: Comparación *frameworks* para aplicaciones de escritorio
Fuente: http://tangiblejs.com/posts/nw-js-and-electron-compared-2016-edition

Como conclusión, decir que emplear los servicios *cloud* en los entornos locales, como aplicaciones de escritorio y con el enfoque propuesto, ofrece la oportunidad de desarrollar un sistema autónomo e independiente de las conexiones de red, modo *offline*. Así se logra conceder al usuario la ejecución del sistema en su equipo y que mediante los mecanismos adecuados pueda efectuar funciones como vincular o exportar sus datos, con la finalidad de relacionarlos y gestionar su información con la aplicación en la nube. Pero para esta tarea hay que investigar cómo crear un entorno privado que cuente con los recursos necesarios para alcanzar el propósito.

3.5.2. Tipos de *cloud* y almacenamiento de la información

Los entornos *cloud* están de moda y está tendencia es digna de estudio y experimentación. Las tecnologías en la nube han permitido la creación de diferentes entornos de trabajo dentro de este contexto, pero lo que interesa en esta sección son los tipos de *cloud* en relación a la gestión de los datos que se depositan en ellos, y al mantenimiento y propiedad de los recursos del servicio [90]. Siguiendo este aspecto se pueden distinguir varios enfoques de *cloud*:

- Nubes privadas, donde la infraestructura con los recursos es utilizada bajo demanda según las necesidades del momento. Estos entornos permiten alcanzar mayores niveles de seguridad y privacidad, debido a que los elementos internos son propiedad de un único cliente. Además, al ser servicios individuales la personalización es posible. La pertenencia de los recursos otorga el control y administración total, pero de igual forma los costes son un poco más altos.

- Nubes públicas, las cuales son administradas y mantenidas por terceros. En este ámbito los recursos de la infraestructura se comparten entre los datos y procesos de diversos usuarios. Todos los elementos del servicio le pertenecen a la empresa proveedora y ésta es quién ofrece las aplicaciones a través de Internet. Dentro de este tipo, se han incluido ciertas características como las *VPN*, que han permitido desarrollar contextos más seguros y que se han denominado con un nuevo término, *Cloud Privado Virtual (Virtual Private Cloud, VPC)*. Los costes relacionados con la adopción de estos entornos son más bajos en comparación con los otros.

- Nubes híbridas o mixtas, siendo el producto de las dos anteriores. Los recursos del entorno son en parte compartidos y otros en propiedad. Esta división de los elementos es reconocida y se tiene bastante diferenciada. Se ofrece a los clientes poder tener su propia infraestructura, pero que cuando necesitan más recursos, éstos pueden ser añadidos a sus requerimientos sin dificultades. No obstante, esta flexibilidad viene contrarrestada por el problema en la gestión de la distribución.

Sobre estas divisiones existen muchos tipos de servicios de almacenamiento, y los interesantes para este trabajo son los gratuitos o los de administración propia. Igualmente, se buscó servicios que pudieran suministrar otro tipo de funcionalidades favorables para la creación de la aplicación. Son de sobra conocidas las ventajas de los ámbitos *cloud*, pero la facilidad de acceso desde cualquier lugar que tenga disponible el sistema y una conexión a Internet, es un hito destacable en la facilidad de uso. Con todo ello, se estudiaron servicios [91] tales como *Google Drive*, *Dropbox*, *OneDrive*, *Mega* y *Box*, pero se descartaron debido a que se perseguía un mayor control en la administración y estructura de la información. Así surgió la opción de emplear el alojamiento web en un servidor (*hosting*) [92] para poder realizar la labor de almacenamiento. Este enfoque permitiría alcanzar las ideas establecidas, además de ofrecer más personalización y libertad en la instalación de recursos. Por razones de proximidad con los lenguajes web y por aumentar los conocimientos de los mismos en otros contextos, *backend* o lado del servidor, la tecnología que se pretende manejar es *Node.js*. También es importante recalcar que su gran diversidad de módulos y características hacen de este *framework* una herramienta ideal para el desarrollo de aplicaciones, facilitando su creación y no delimitándolas por cuestiones técnicas.

Pero después de pensar cómo realizar el almacenamiento de los datos en un servidor remoto, se encontró la herramienta *Firebase* [93], la cual ofrece una gran cantidad de características útiles y recursos *backend* como servicios para el desarrollo. Inclusive, tiene una excelente documentación, donde encontrar la información específica de sus recursos. Entre sus funcionalidades la que se destacó fue la base de datos en tiempo real, aunque también se emplearon otros módulos como el de autenticación. Aspectos favorables de *Firebase* son la flexibilidad de uso y la compatibilidad con los entornos web. La base de datos maneja la información con una estructura *JSON* y es del tipo de bases de datos *NoSQL*. Además, la sincronización y almacenamiento de la información se

efectúa de manera veloz, lo que permite ofrecer bajos tiempos de respuesta para los usuarios finales. Por otro lado, de la característica de autenticación se hablará en la sección correspondiente a la seguridad y privacidad.

En último lugar, también hay que mencionar que como se pretende ofrecer una aplicación que se pueda utilizar en un contexto local sin la necesidad de usar recursos de Internet, se empleó una base de datos en el lado del cliente para guardar la información del usuario. Más concretamente, se optó por una base de datos *NoSQL, IndexedDB*, debido a que era del mismo tipo que la del servicio *Firebase* y que este género de bases de datos no se había empleado antes, por lo que también supuso una inquietud de conocimiento. Además, para interactuar con esta base de datos local se decidió trabajar con el *framework NeDB* [94].

3.5.3. Seguridad y privacidad

Con la firme intención de dotar al sistema de procesos y mecanismos apropiados para realizar la gestión de la información del calendario de los usuarios de modo seguro y privado, se analizaron los diferentes métodos y algoritmos de cifrado para concretar cuál o cuáles son los más convenientes y eficaces para cumplir con esta labor. La idea es realizar todas las operaciones criptográficas en el lado del cliente, para que una vez se almacenen los datos, ya sea en un servidor externo no confiable o en la base de datos local propia, los contenidos estén protegidos y no sean legibles por cualquier cosa o persona no autorizada.

Por lo anterior mencionado, se comenzó con la investigación. La criptografía [95], [96] emplea métodos de cifrado a través de algoritmos y claves para alterar la forma de los mensajes, y que éstos se conviertan en textos no comprensibles, en un principio. Atendiendo a las claves que se manejen en los procesos criptográficos, se pueden diferenciar tres tipos de cifrado:

- Simétrico o de clave privada/secreta/única: Este tipo de cifrado utiliza con el algoritmo una misma clave para las tareas de cifrado y descifrado de los mensajes. Esto resulta un problema, debido a que o se tiene o se consensua la clave a emplear previamente entre los actores, o el proceso de distribución de la clave, siempre que sea necesario, va a ser complejo de llevar a cabo. Además, esta complicación se agrava, si la compartición de la clave es vulnerada, ya que si ésta cae en manos inapropiadas las consecuencias son

incalculables. Pero las ventajas que hacen que este mecanismo se use es que es bastante rápido en la realización de los procesos vinculados a la encriptación y a que la longitud de las claves no es demasiado grande para funcionar correctamente. Los modos más relevantes de este tipo de cifrado son: el *DES*, el *Triple DES* y el *AES*.

- Asimétrico o de clave pública: Al contrario que el método de cifrado anterior, este tipo emplea con el mecanismo criptográfico un par de claves distintas. Estas claves son diferenciadas en clave pública y clave privada. La primera se utiliza para cifrar los contenidos y la segunda sirve para recuperar la información original mediante el descifrado de los datos. La criptografía asimétrica tiene unas características importantes en sus claves y son, que su clave privada no se puede inferir, en un contexto práctico, a través de la clave pública, y que el conjunto de claves, pública y privada, únicamente se puede generar una sola vez. Además, también hay que destacar que elimina el inconveniente de la compartición segura de las claves, pues en este caso la clave a transmitir es la pública. Igualmente, otra propiedad útil es la posibilidad de firmar los mensajes con este método. Sin embargo, no todo son ventajas, pues el coste computacional y los tiempos que se manejan en estos procesos son altos. Y mencionar que el tamaño de las claves, para que éstas sean efectivas, debe ser bastante largo. Con la intención de remediar esta problemática, se están incorporando a estos modelos de cifrado las *curvas elípticas*. Los algoritmos más destacados del cifrado asimétrico convencional son: el *RSA* y el *ElGamal*.

- Híbrido: Esta forma de cifrado pretende aunar las ventajas de los dos modos previos e intentar solucionar sus dificultades. Dependiendo de las necesidades a suplir se usan en un contexto, las claves simétricas o las claves asimétricas. La primera de éstas se utiliza para la distribución de los datos, mientras que la segunda se usa para el intercambio de la clave del ámbito de la criptografía simétrica entre los elementos presentes en la comunicación. Más detalladamente, el proceso seguido en este modelo es: Primero se genera una clave de sesión para el mensaje a transmitir. Esta clave es de tipo simétrico. Después, la misma se aplica en el cifrado del contenido del mensaje y, además, es encriptada con la clave pública del receptor de la información. Tras ello, se combinan ambas partes en un sólo paquete y éste es enviado. Cuando se recibe este elemento, el destinatario con su clave privada

consigue la clave de sesión incrustada en el paquete. Con ella, ya se puede descifrar el resto del contenido, y así obtener la información que se pretendía comunicar. Dentro de este tipo de cifrado se tienen dos herramientas importantes que hacen uso del modelo, éstas son: *PGP (Pretty Good Privacy)* y *GPG (GNU Privacy Guard o GnuPG).*

Los diferentes tipos de cifrado conviven unos con otros, ya que ninguno sustituye a otro, debido a que cada uno de ellos tiene su ámbito de utilización y sus puntos favorables. Por ende, los métodos de cifrado son una buena manera de suministrar seguridad y privacidad a la información, pero no es suficiente, ya que también hay que tener en cuenta la integridad de los contenidos y la autenticación de los elementos relacionados con el mensaje. Debido a esto, se hace necesario tratar con diferentes alternativas presentes en la criptografía o con distintas maneras de usar los cifrados, para intentar ofrecer en la aplicación un entorno lo más completo posible en estos aspectos. Además, en el mundo de la seguridad la máxima que se rige es que "*un método criptográfico es bueno si la seguridad recae totalmente sobre la clave y no sobre el algoritmo de cifrado empleado*" [97].

Tras examinar y ver los mecanismos que se ofrecen para el desarrollo en los aspectos relativos a la seguridad y la privacidad, se decidió emplear el enfoque de la criptografía híbrida para gestionar la información de los eventos de los usuarios. Después de efectuar la elección, se buscó la manera de encontrar una relación con las tecnologías web [98], y se halló el protocolo y estándar *OpenPGP* [99], [100], el cual es bastante importante para la creación de la aplicación, pues proveerá los mecanismos y procesos oportunos para realizar la administración, verificación e identificación de los recursos personales de los usuarios. Es más, hay que destacar que no está sujeto a ningún sistema propietario, es decir, el estándar *OpenPGP* es de código abierto y libre de licencias. Esta estructura es frecuentemente utilizada entre las personas que pretenden preservar su información, por esta razón algunas organizaciones suministran servidores donde poder acumular las claves públicas generadas. Esto fue de interés, pero finalmente se decidió llevar a cabo esta actividad dentro de los recursos de almacenamiento propios. La finalidad de las claves públicas en la aplicación es la compartición de los eventos del calendario entre los usuarios asistentes. En efecto, los eventos se cifran con una clave AES de sesión y esta clave se distribuye mediante su cifrado con la clave pública de cada uno de los usuarios con los que se desea compartir eventos. Después,

el evento cifrado se envía a las cuentas de los asistentes. Cuando los usuarios del evento intenten obtener la información del recurso, necesitarán su clave privada para recuperar la clave AES y descifrar el fichero con el evento. Además, la información de los usuarios y otros metadatos son cifrados mediante AES antes de guardarla de modo persistente, bien en la base de datos local, bien en la base de datos remota.

Por otro lado, para añadir a la aplicación un sistema de acceso basado en la identificación de los usuarios, se usó el módulo de autenticación del servicio *Firebase*, como ya se había mencionado en un apartado anterior. Éste ofrece unas características muy adecuadas para el desarrollo, como la sencillez, tanto en el manejo para el usuario final como en la implementación para el programador, y la robustez en la gestión de los datos. Asimismo, el módulo es gratuito y multiplataforma. La forma de emplear esta herramienta fue mediante la creación de cuentas de usuario con nombre, correo electrónico y contraseña. Los dos campos finales eran los que se utilizaban para la autenticación. Igualmente, el módulo ofrece otros mecanismos de identificación, pero no se aprovecharon ya que no se vio la necesidad, aunque pueden ser tenidos en cuenta para mejoras futuras.

El mecanismo descrito previamente se dispuso para la parte Web del sistema. En el modo de funcionamiento local la gestión de usuarios se realiza contra la base de datos local y de modo independiente de la gestión de usuarios del servicio remoto. Así, un usuario puede crear varios perfiles locales mediante la correspondiente dirección de correo electrónico y una contraseña. Estas credenciales son almacenadas en la base de datos local, de forma que una vez autenticado un perfil se crean y gestionan eventos de calendario en local. Dichos eventos no son sincronizados con los eventos almacenados en *Firebase*. Si se desea realizar esta operación, la herramienta desarrollada permite exportar la información desde la base local e importarla en el servicio *cloud*.

3.5.4. Estructuras de datos

La organización de la información dentro de ciertos elementos escogidos para formar parte del sistema viene delimitada por el servicio o el estándar. No obstante, todavía se tienen que seleccionar algunas estructuras para representar y guardar los datos de la aplicación de escritorio, tanto relativos a los usuarios como los que están vinculados con los calendarios.

Primero, se estudió la forma de manejar la información de los eventos del calendario de los usuarios. Para no complicar las cosas y ofrecer la interoperabilidad entre plataformas del mismo estilo, se eligió como estructura para los eventos el estándar libre más extendido y estable, *iCalendar* [101], [102]. En este formato de entre sus características se puede resaltar la flexibilidad en la personalización, pues aparte de contar con una organización fija y marcada, da la posibilidad de insertar nuevos componentes rápidamente y sin complicaciones. Con esta distribución serán almacenados los recursos de los calendarios en las bases de datos y serán generados los ficheros que contengan información de eventos, siempre y cuando, el contenido se tenga que presentar de modo claro o legible.

Después, se buscó un método de representación que permitiera ofrecer al usuario un mecanismo de obtención de su información ágil y sencillo. Los datos a exportar serían parte de la información personal e incluso eventos del calendario. Esto se persigue con la intención de aumentar las funcionalidades de la aplicación y maximizar la experiencia del usuario. Aquí se hizo uso de los códigos *QR (Quick Response Code)* [103]. Estos elementos son unos códigos de barras bidimensionales o matriz de puntos con un formato cuadrado, que se pueden personalizar para contener diferentes tipos de información como, por ejemplo, direcciones *URL* o texto. Este tipo de representación de la información se puede distinguir por su estructura con cuatro lados iguales y por tener en su interior tres cuadrados más pequeños alojados cada uno de ellos en los vértices, exceptuando la esquina inferior derecha. Este formato consta de derechos de propiedad, pero no se aplican, por lo que es completamente libre y sin regalías. Igualmente, está estructura queda definida por un estándar que especifica sus particularidades. Además, estos códigos pueden ser escaneados a través de los dispositivos móviles que dispongan de una cámara o en los equipos que puedan incorporar un lector de códigos de barras. También es imprescindible una herramienta apropiada que realice la decodificación de la información. Estos elementos dan la posibilidad de ejecutar acciones en los dispositivos, como enviar un mensaje, realizar una llamada o almacenar un evento en la agenda. Este aspecto de interconexión entre diferentes dispositivos y entornos aporta cierto grado de innovación que, si bien no es del todo novedoso, si es algo que no se suele ver en las aplicaciones de gestión de eventos de calendario. Para finalizar, los códigos *QR* en el sistema serán generados mediante el uso del lenguaje *JavaScript*.

4. Diseño

En este apartado se va concretar la organización, estructura y diseño de la aplicación de escritorio para la gestión del calendario del usuario. Igualmente, se muestra y detalla el flujo de navegación entre los componentes del sistema. Asimismo, se especifica el esquema que va a tener la información dentro de las bases de datos. Todos estos elementos deben seguir las características marcadas en el análisis realizado previamente.

4.1. Definición del diseño

Esta fase es de vital importancia, debido a que una correcta especificación del diseño del sistema a implementar hace que se pueda obtener una aplicación de calidad e idónea para su utilización por los usuarios. La estructura del sistema a diseñar debe cumplir con aspectos tan importantes como la escalabilidad, la mantenibilidad y la facilidad de desarrollo, tanto actual como futura. Por ello, la aplicación se basa en una disposición del código en módulos, con la intención de delimitar las funcionalidades de cada uno de ellos. Así se facilitaría la posibilidad de aprovechar el código generado como el modo de poder integrar nuevas características. Esto apoya las pautas de que las funciones y módulos de la herramienta *software* a crear deben alcanzar la mayor cohesión y el mínimo acoplamiento posible [104]. Igualmente, no es menos importante realizar una buena documentación y trazabilidad de los elementos presentes en los archivos fuente de la aplicación.

El sistema en concreto se crea con la idea de suministrar a los usuarios y a la comunidad, una herramienta que sea capaz de gestionar los eventos de calendario personales, en la nube o de forma local, cumpliendo ciertos aspectos de seguridad y privacidad. Además, la aplicación tiene el enfoque de emplearse en entornos de escritorio, independientemente de cual sea el sistema operativo que esté ejecutando el equipo donde se usa el servicio. Como se están utilizando las tecnologías web para acometer el desarrollo y la aplicación debe satisfacer estas características descritas, las interfaces deben tener un diseño cuidado y bien pensado para que todas las funcionalidades del sistema sean útiles, y que la experiencia de usuario no se vea afectada por

cuestiones técnicas. Los requisitos específicos de las interfaces para que se pueda lograr el objetivo de parecer una aplicación de escritorio nativa y cumplir con ciertos niveles usabilidad son:

- Estructuración: Las pantallas deben tener una organización y distribución correcta en los límites definidos de trabajo, ocupando el espacio suficiente y necesario para maximizar la experiencia de uso.

- Consistencia: Los elementos presentes en las interfaces de la aplicación deben tener las mismas pautas de diseño, tanto en formato como visuales. Esto otorga coherencia y minimiza las dificultades de comprensión. Asimismo, la estructura debe ser igual en los diferentes entornos de ejecución.

- Uniformidad: Las acciones y flujos de transición de las pantallas deben ser lo más ecuánimes posibles, y más si se encuentran en el mismo ámbito de trabajo. Además, los componentes deben favorecer la interrelación y la localización de los mismos.

- Intuitiva: Los elementos y funciones mostrados en el sistema deben revelar sus características de modo directo, conciso y preciso, para no ofrecer un entendimiento erróneo o equivocado al usuario. Los componentes se deberían explicar por sí mismos.

- Sencillez: Las interfaces no deben ser demasiado cargadas para minimizar la complejidad y favorecer la usabilidad. Así se logra mayor adaptación y menor requerimiento de habilidad y conocimiento por parte de los usuarios finales. Los mecanismos y funcionalidades presentados deben ser fáciles y requerir únicamente la puntual intervención de un sujeto externo.

- Informativa: Las pantallas deben suministrar mensajes de información a los usuarios. Éstos deben ir relacionados con las acciones que realice el usuario en la aplicación. El contenido de la información debe ser claro y directo. No obstante, la utilización de los mensajes no se debe exceder de unos límites apropiados, pues mostrar una cantidad elevada de ellos, puede resultar molesto y deteriorar la experiencia en el uso del sistema.

- Tiempo de respuesta bajo: La aplicación tiene que ofrecer una navegación y realización de las operaciones rápida y fluida. Las acciones llevadas a cabo en el sistema no deben sobrepasar unos tiempos prudentes y lógicos en función del trabajo a ejecutar.

Otros aspectos del diseño que también son importante están vinculados con la forma de entender la seguridad y la privacidad de los datos manejados en la aplicación. Así, estos puntos tienen que ser analizados y tenidos en cuenta en el diseño, porque, aunque se pueden añadir al finalizar el desarrollo de las herramientas *software*, esto no es una buena idea. Pues a pesar de agregar estas características, no se suelen pensar debidamente, y por ello, pueden ser la causa de la aparición de errores y vulnerabilidades. Por eso la forma más correcta de abordar este tema, es asegurándose que la seguridad y la privacidad son un contenido fundamental en el diseño de la aplicación, siguiendo los principios de *security and privacy by desing* [105], [106].

Debido a todo ello, en el sistema de escritorio a implementar se van a utilizar el cifrado simétrico e híbrido para suministrar estos requisitos. Concretamente, se va a trabajar con el primer método para proteger toda la información de los usuarios en las bases de datos, y el segundo cifrado se va a usar para manejar los eventos del calendario. Con este proceso se consigue que los datos permanezcan seguros y privados, pues todos los recursos personales son encriptados en el lado del cliente, antes de guardar los elementos en las bases de datos, remota o local.

Esta forma de proceder dotaba a la aplicación de tres conceptos relevantes en los entornos de la seguridad web. Los fundamentos en los que se asienta el diseño son:

- *Client-side encryption* [107]: Es una técnica criptográfica que se basa en el cifrado de la información en el emisor antes de enviarla al servidor remoto a través de Internet. Esto se hace para asegurar los datos y que solamente sean comprensibles en el lado del cliente mediante el descifrado de la información con la clave del usuario. Como contrapartida, el usuario ha de asumir tanto la carga del cifrado de la información como la custodia de las claves de cifrado.

- *Zero-knowledge server* [68]: Es un enfoque, derivado de la utilización de la anterior característica, en el cual el servidor empleado para gestionar la información de los usuarios no tiene conocimiento sobre los recursos, siendo casi imposible que se pueda obtener el

contenido de los datos, tanto para los servidores como para terceros, pues ninguno de ellos han sido autorizados. Esto se emplea porque es mejor no confiar en los servidores ni en los proveedores de los servicios de Internet.

- *End-to-end encryption (E2EE)* [108]: Esta característica está fuertemente relacionada con los dos elementos previos. Este modelo de comunicación permite que únicamente los usuarios que intervienen en la misma, puedan obtener la información de los mensajes intercambiados. Esto es de utilidad en el contexto de la compartición de eventos de calendario entre los usuarios asistentes que se encuentran en la aplicación.

4.2. Estructura de la aplicación

En esta sección se describe la organización a adoptar en el proyecto *software*. Debido a que para realizar la implementación de la aplicación de escritorio se va a utilizar el *framework Electron*, éste ya define una estructura adecuada para la distribución de los ficheros. Se va a tener una carpeta principal que es la raíz del proyecto. Como los sistemas creados con esta herramienta constan de dos procesos destacados, el principal y el de *renderización*, hay una separación evidente. El primero se encuentra en el fichero *JavaScript, main.js*, mientras que el segundo se halla en una carpeta para la aplicación, *app*. Además, todos los módulos y dependencias de desarrollo a manejar se almacenan en la carpeta *node_modules*. Todos los detalles de la configuración están contenidos en el archivo *package.json*. Dentro de la carpeta del sistema, ya no hay una estructura específica a cumplir, y se siguió una organización basada en módulos, donde la diferenciación era su ámbito de funcionamiento en la aplicación. Por ello, en este contexto se tienen las siguientes partes:

- *HTML*: Consta de un solo fichero, el cual es *index.html*. La razón de esto es que se está empleando un patrón de diseño *SPA*. Este archivo es la entrada de ejecución del proceso de *renderizado*. El nombre designado es porque se están empleando tecnologías web, y hay unos estándares que se intentan seguir.

- *CSS*: Los ficheros de estilos se agrupan en una carpeta denominada *css*. En ella se alojan los archivos que se encargan de suministrar y definir las características visuales y de

distribución de los elementos de la interfaz de la aplicación. Se tiene un fichero principal, *styles.css*, que son los estilos del sistema, y una carpeta *themes*, con los temas ofrecidos al usuario para la personalización de las pantallas de la herramienta *software*.

- *JavaScript*: Los archivos de funcionalidad se localizan en una carpeta llamada *lib*. Ésta contiene los ficheros *JavaScript* de la aplicación de escritorio. Los recursos que se establecieron en este contexto se distinguieron según las funciones que iban desempeñar en el sistema. Así se tienen los siguientes módulos:

    - Calendario: Consta de un fichero nombrado *calendar.js*. En este recurso se depositarán las funciones principales de administración del calendario y de la aplicación.

    - Seguridad: Tiene un archivo llamado *crypto.js*. Este elemento contendrá los métodos y procesos criptográficos que aportarán la seguridad y la privacidad a los datos manejados en el sistema.

    - Datos: Consta de un fichero denominado *ical.js*. En este fichero fuente se implementarán las funciones para gestionar la información relacionada con los eventos *iCalendar*.

    - Almacenamiento: Tiene un archivo designado como *storage.js*. Este recurso recogerá las funciones encargadas de persistir la información en las bases de datos, tanto local como remota.

4.3. Estructuras de datos

Con el propósito de gestionar la información de los eventos del calendario personal de los usuarios en la aplicación mediante las tecnologías web, en concreto con *JavaScript*, se definió una estructura de datos para los objetos que tienen que cumplir con esta tarea. Ésta consta de los siguientes parámetros: identificador, título, fecha de inicio, fecha de fin, fecha de creación, fecha

de última modificación, *URL*, lugar, descripción, secuencia, color, indicador de evento diario u horario, indicador de evento disponible u ocupado, organizador del evento y asistentes al evento.

Por otro lado, también hay que tener en cuenta la organización de la información a guardar en las bases de datos. Ésta debe poseer una estructura adecuada para poder ordenarla de forma correcta y que las operaciones a ejecutar sobre la misma se realicen eficientemente y sin errores. Por ello, la estructura de datos para almacenar la información de los usuarios está compuesta por los siguientes campos: identificador, nombre, correo electrónico, contraseña, clave pública, calendario, lista de eventos compartidos (identificador, organizador y evento) y lista de eventos por compartir (identificador, asistentes y evento).

4.4. Flujo de navegación

En esta sección se representa y expone el flujo de navegación (Ver Figura 8) entre los diferentes estados que tiene la interfaz de la aplicación de escritorio. Al tratarse de un sistema basado en el diseño *SPA*, no hay transiciones entre pantallas, pero sí hay un intercambio de estados entre los componentes.

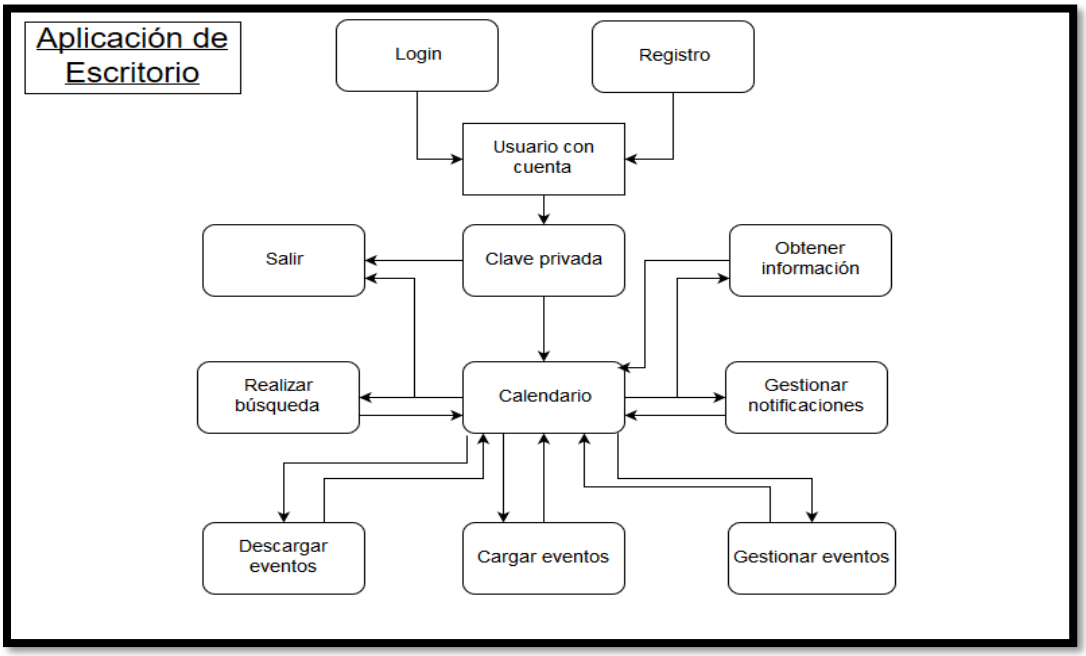

Figura 8: Diagrama del flujo de navegación de la aplicación de escritorio

5. Implementación

En este apartado se detallan las actividades y los mecanismos que se emplearon para el desarrollo de la aplicación de escritorio. Igualmente, se especifican y explican los entornos de trabajo, las herramientas y aplicaciones utilizadas en todo el proyecto, los ficheros fuente generados y el proceso para la elaboración de la documentación.

5.1. Entorno y herramientas de trabajo

A consecuencia de la naturaleza del proyecto, los recursos necesarios para su desarrollo no son elevados. De todas maneras, también se intentó reducir lo máximo posible los costes de producción de la aplicación *software*, eligiendo herramientas libres y gratuitas o que estuvieran disponibles mediante la intervención de la universidad. En primera instancia, se va a proceder a exponer el entorno de trabajo donde se ha llevado a cabo la creación del sistema y todas las tareas que en este proyecto se tratan. El equipo *hardware* del que se dispuso fue un ordenador personal de escritorio. Esta máquina tiene los siguientes componentes y sus características técnicas son:

- Procesador: Intel i7-6700K 4.0Ghz.
- Memoria RAM: 8 GB.
- Disco Duro: SSD 500 GB.
- Sistema operativo: Windows 10 Pro.
- Tarjeta gráfica: NVIDIA GeForce GTX 1070 8GB.
- Monitor: LG 29 pulgadas, formato 21:9 y resolución 2560x1080 píxeles.

También hay que mencionar que aparte del sistema operativo principal, *Windows*, se empleó un entorno *virtualizado* con un sistema operativo *Ubuntu* 16 de 64 bits para verificar el funcionamiento de la aplicación en diferentes entornos y con la finalidad de solucionar posibles errores de implementación.

Tras comentar el entorno *hardware*, ahora se van a detallar las herramientas *software* que se han usado, tanto para crear el sistema de escritorio como para facilitar el trabajo del proyecto. Las aplicaciones y servicios utilizados son:

- *Visual Studio Code (VS Code)*: Este programa es un editor de código desarrollado por Microsoft y basado en el mismo *framework* empleado para crear la aplicación del proyecto, *Electron*. Esta herramienta permitió la implementación y ejecución del sistema, manejando correctamente todos los archivos de los lenguajes web. Es un entorno multiplataforma, libre y gratuito. Además, integra componentes para realizar un adecuado control de versiones.

- *VMware Workstation*: Este programa facilita la tarea de crear entornos *virtualizados* con distintos sistemas operativos. Su utilización permitió definir un entorno donde poder realizar pruebas en otros equipos.

- *Bitbucket*: Es un servicio de alojamiento *cloud* que suministra funciones para añadir un sistema de control de versiones en los proyectos informáticos. Esta herramienta ofrece de forma gratuita la posibilidad de tener repositorios privados, donde poder almacenar las aplicaciones.

- Consola de *Firebase*: Esta herramienta es el sistema de depuración del servicio *Firebase*. Con ella se pueden realizar diferentes operaciones sobre los módulos de la base de datos o de autenticación.

- Consola del navegador web: Al desarrollar el sistema mediante *Electron*, el programa generado se basa en el motor de *Chrome* para su ejecución. Por ello, se tiene acceso a la consola de desarrolladores, pudiendo así realizar la depuración de los elementos de la aplicación, ficheros *HTML*, *CSS* y *JavaScript*.

- *QUnit*: Este *framework* ofrece la posibilidad de realizar test unitarios a las funciones *JavaScript*. Dicha tecnología se usó para validar las funciones de ese contexto que se crearon en la aplicación.

- Lector *QR*: Es una aplicación de dispositivos móviles que se emplea para interpretar los códigos *QR*. Se utilizó para obtener la información generada por la aplicación.

- *iCalendar Validator*: Es un servicio *online* que sirve para comprobar que las estructuras con formato *iCalendar* están correctamente creadas.

- *Mendeley*: Esta aplicación ayuda en la gestión de las referencias. Es un programa gratuito y se empleó para la documentación de la bibliografía del documento final.

- *JSDoc*: Esta herramienta sirve para crear documentación técnica de los ficheros fuente con código *JavaScript*, y se utilizó con este único propósito.

- *Microsoft Office*: Esta solución informática sirve para diversas tareas de ofimática, pero las relevantes en este proyecto son la generación de documentos de texto y presentaciones. Estas herramientas contribuyeron con la redacción de la memoria y la elaboración de las diapositivas.

- *Adobe Reader*: Este programa suministra la función de mostrar el contenido de los ficheros con extensión *pdf*. Esta herramienta es de gran ayuda para el análisis y lectura de la documentación requerida.

- Buscador de Google: Este servicio es una de las aplicaciones más usadas en todo desarrollo de cualquier proyecto, pues permite obtener diversa información efectuando búsquedas en Internet. Su utilidad es de sobra demostrada, y facilita la resolución de problemas al ser la herramienta de contacto para llegar a la información o a la comunidad.

5.2. Codificación

Después de conocer todas las características del entorno de trabajo, en esta sección se va a detallar el código de los diferentes módulos creados. La aplicación de escritorio multiplataforma se implementó según las pautas definidas en las fases de análisis y diseño. Para realizar una explicación organizada, se van a describir los ficheros generados según el contexto o módulo al que pertenecen. Por esta razón, la exposición es la siguiente:

- Principal

En este contexto se halla el fichero *main.js* (Ver Figura 9). En este archivo se declaran las características y funciones de la ventana de la aplicación de escritorio. Este recurso es el que permite ejecutar la aplicación web en el entorno de escritorio local. Entre sus funcionalidades está la personalización y la capacidad de realizar acciones sobre la ventana de trabajo del sistema.

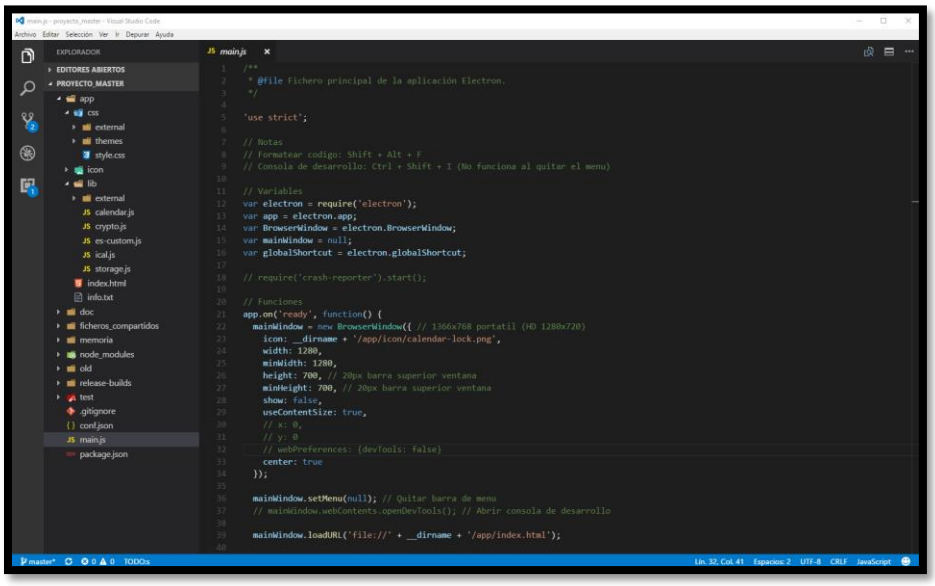

Figura 9: Fichero main.js

- Estructura

En este contexto se encuentra el archivo *index.html* (Ver Figura 10). En este fichero se añadieron los elementos y componentes estructurales de la interfaz de la aplicación. Gracias a las etiquetas semánticas de *HTML5*, se pudo organizar los contenidos de forma más adecuada. La disposición de los recursos en el interior se hizo con el objetivo de aumentar la usabilidad y la experiencia de usuario. La estructura de la interfaz consta de tres componentes destacados: el principal de ellos

es el calendario. En esta sección se localizan los componentes que muestran la información del estado de la obtención de eventos. También están los que permiten gestionar las notificaciones de eventos compartidos, y los que representan el calendario principal. Otro de los elementos importantes es la cabecera. Ésta incorpora los recursos para gestionar el estado de la barra lateral, realizar búsquedas de eventos, mostrar el estado de la conexión y publicar las opciones del usuario. Y el último componente de los tres mencionados es la barra lateral. En ella, se hallan los elementos que permiten la navegación, personalización y gestión del calendario. Igualmente, en la aplicación también se encuentran los recursos que dan la información del sistema y los elementos de los entornos de administración de las tareas de acceso, clave de usuario, búsqueda, notificaciones, gestión de eventos y, carga y descarga de eventos.

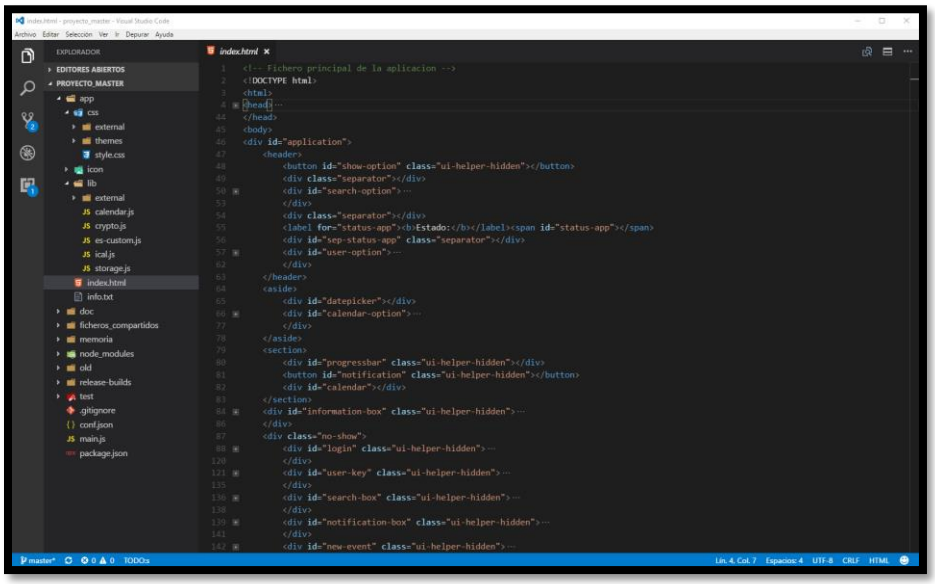

Figura 10: Fichero index.html

- Diseño

En este contexto se localiza el fichero *style.css* (Ver Figura 11). En este archivo se especificaron las características para la apariencia y posición de los componentes estructurales de la interfaz de la aplicación. Asimismo, se incorporaron las propiedades idóneas para garantizar la adaptabilidad de la interfaz a las diferentes resoluciones de pantalla.

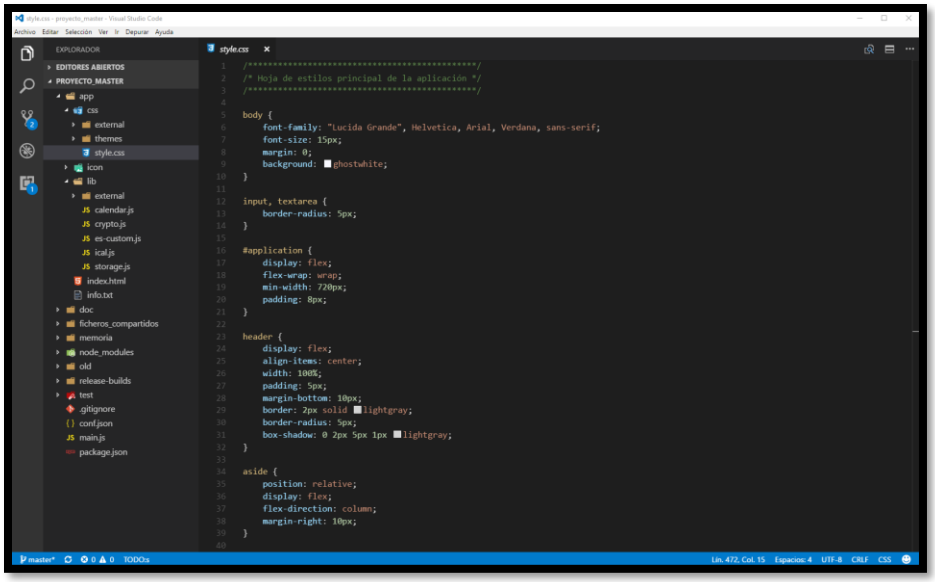

Figura 11: Fichero style.css

- Funcionalidad

En este contexto se engloban los cuatro ficheros de los módulos de la aplicación. A continuación, se detallan los archivos según esta designación.

- Calendario

En este módulo se sitúa el fichero *calendar.js* (Ver Figura 12). En este archivo se implementaron las funciones que se encargaban de toda la gestión del calendario de la aplicación. Del conjunto total de las funciones se pueden destacar las que se emplean para la administración de los usuarios y eventos del calendario, la gestión de la información de los formularios, el manejo del calendario, la gestión de procesos, la búsqueda de eventos, la administración de la compartición de eventos y sus notificaciones, la carga y descarga de datos del usuario, la gestión de operaciones críticas y, las de saneamiento y limpieza del sistema.

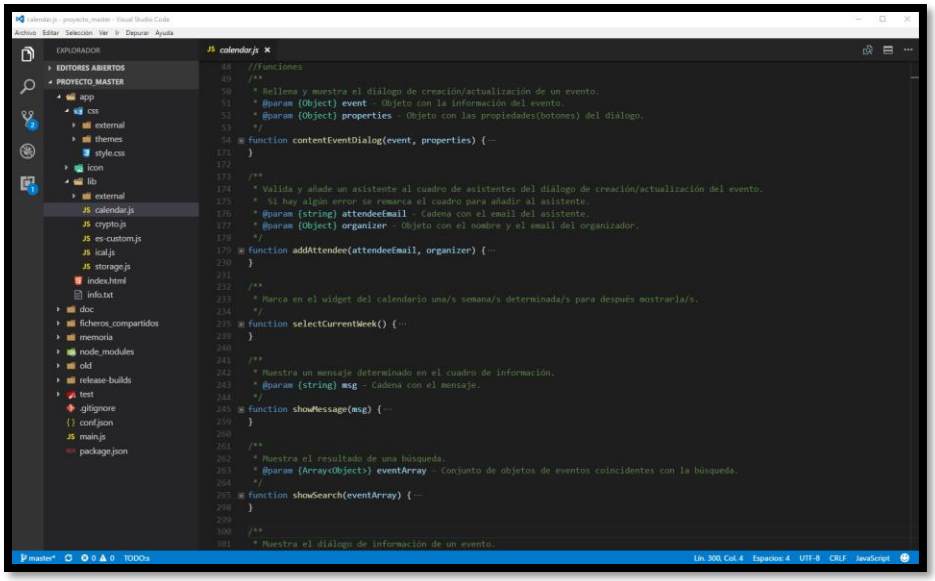

Figura 12: Fichero calendar.js

- Seguridad

En este módulo se localiza el archivo *crypto.js* (Ver Figura 13). En este fichero se codificaron las funciones que suministraban los mecanismos y procesos de seguridad. Entre estas funciones se encuentran las que realizan la generación de claves, la validación de claves privadas y, el cifrado y descifrado de los calendarios e información de las bases de datos.

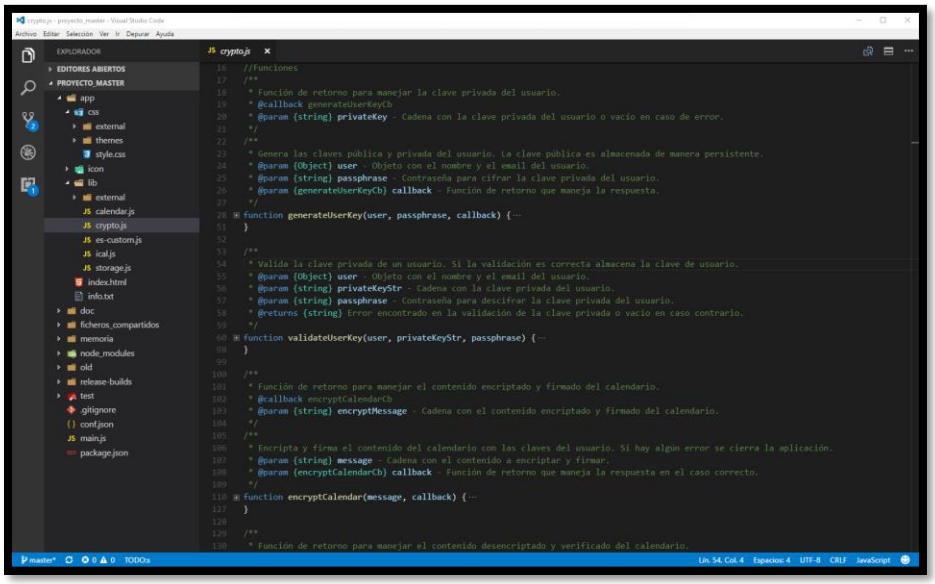

Figura 13: Fichero crypto.js

- Datos

En este módulo se encuentra el fichero *ical.js* (Ver Figura 14). En este archivo se escribieron las funciones vinculadas al estándar *iCalendar*, y que trataban la gestión de la estructura del estándar, la normalización de componentes, la generación de los datos, la conversión de los contenidos y la captura de información de una dirección *URL*.

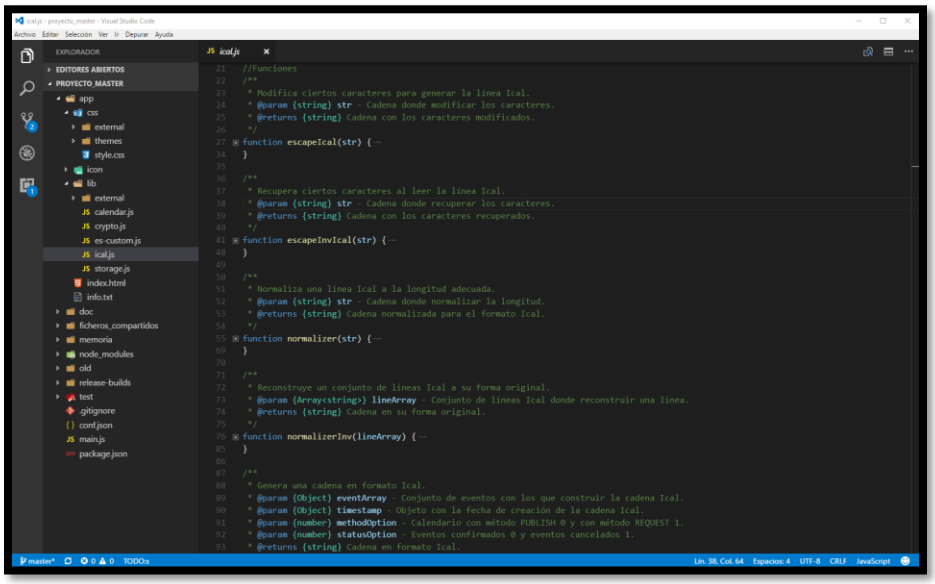

Figura 14: Fichero ical.js

- Almacenamiento

En este módulo se halla el archivo *storage.js* (Ver Figura 15). En este fichero se codificaron las funciones para la gestión de la información del usuario en las bases de datos local y web, y, la autenticación y registro de acceso a la aplicación. Entre las operaciones de la base de datos están crear, obtener, guardar, actualizar, compartir y eliminar información y eventos del usuario.

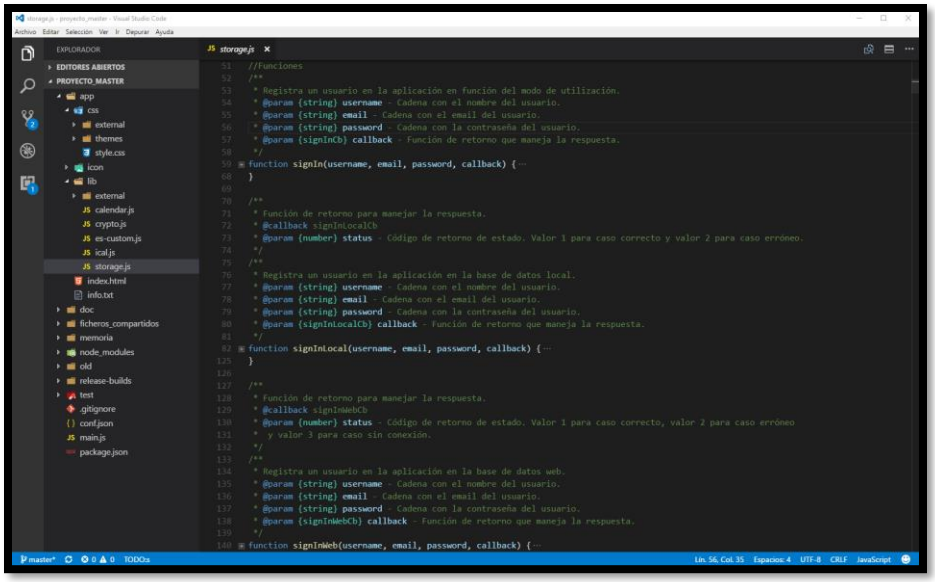

Figura 15: Fichero storage.js

5.3. Documentación

Para concluir la fase de implementación es importante generar una documentación adecuada de la aplicación. Para cumplir con esta tarea correctamente, hay que alcanzar dos hitos. El primero se realiza durante la escritura del código, insertando comentarios apropiados que permitan entender las funciones y seguir el flujo de procesamiento con sencillez, destacando las particularidades con mayor interés de cada método y ámbito codificado. El segundo es elaborar una documentación técnica (Ver Figura 16). Para esta actividad se empleó la herramienta *JSDoc*, la cual ofrece los mecanismos necesarios (configuración mediante el archivo *conf.json* y salida en la carpeta *doc*) para detallar el procedimiento concreto de las funciones *JavaScript* creadas. Esto permite que los contenidos sean comprendidos sin problemas. Este proceso completo, aunque parezca innecesario y laborioso, es de gran utilidad cuando se debe llevar a cabo un mantenimiento o, se quiere conocer o confirmar el funcionamiento de la aplicación, pues favorece que estas acciones se puedan ejecutar de modo rápido y fácil.

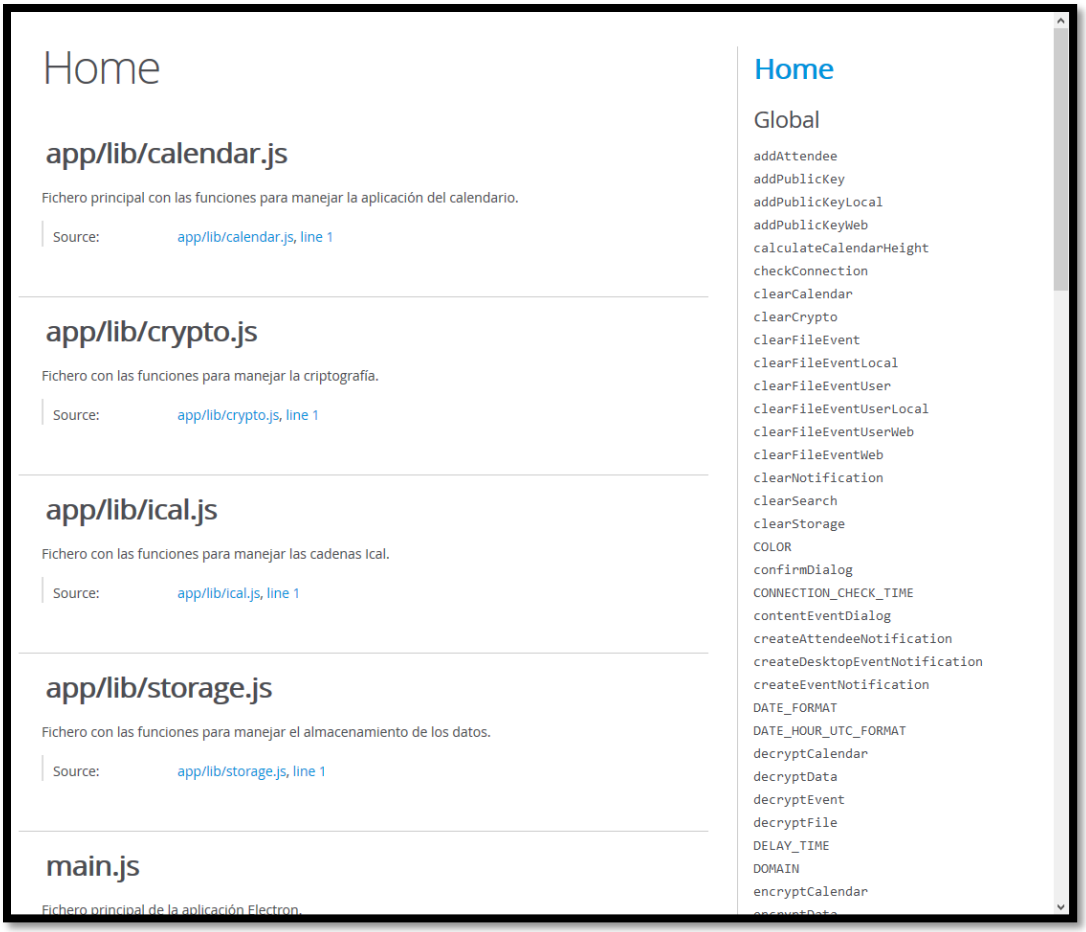

Figura 16: Página de inicio de la documentación técnica

6. Pruebas

En este apartado se especifica y detalla el conjunto de pruebas a realizar para comprobar la completitud del sistema y para conseguir encontrar la máxima cantidad de errores posibles.

El propósito de testear la aplicación es obtener información relevante que permita saber si se cumplen ciertos criterios de calidad *software*. Esto brinda la posibilidad de llevar un control de la herramienta, para ofrecer a los usuarios finales, un adecuado producto que aporte una buena experiencia en su utilización.

Primero, se efectuaron pruebas estáticas para seguir el flujo de procesamiento de la aplicación y verificar si lo que se hizo en la fase de implementación fue correcto. Estas pruebas se formalizan sin la ejecución de ninguna línea de código del sistema. Aunque este tipo de pruebas son en cierto modo útiles, no son del todo eficaces y suficientes. Por ello, también se contemplaron las pruebas dinámicas, con las cuales ya se podía testear la aplicación en funcionamiento.

Las pruebas a realizar en este ámbito, se dividen según la finalidad con la que se apliquen o el contexto específico a testear. Por ello se tienen pruebas unitarias, de integración y del sistema. La intención de cada una de estos tipos de pruebas es:

- Pruebas unitarias: Estas pruebas sirven para verificar el comportamiento de las funciones codificadas de forma independiente y aislada.

- Pruebas de integración: Este tipo de pruebas otorgan la posibilidad de comprobar la correcta y adecuada vinculación de los diferentes componentes y módulos implementados. El propósito es observar la interoperabilidad y la funcionalidad en conjunto.

- Pruebas del sistema: Estas pruebas se usan para experimentar con toda la aplicación desarrollada. El sistema es verificado usando todas sus características y es probado en diversos entornos y situaciones.

Para las pruebas unitarias se empleó la herramienta *QUnit*. Las tareas a realizar fueron ejecutar los métodos que llevaran a cabo procesos atómicos sin ninguna relación con otras funciones. Se crearon módulos paralelos a los ya presentes en la aplicación para poder verificar su funcionalidad en su propio contexto. Estas pruebas permitieron conseguir información relevante para poder tomar decisiones y corregir los primeros errores.

Posteriormente, para las pruebas de integración se definieron nuevas tareas, que está vez sí, relacionan las funciones de los distintos módulos del sistema de escritorio desarrollado. La ejecución entre los módulos para comprobar su conexión adecuada fue útil y necesaria para seguir obteniendo fallos y así subsanarlos. Ya con estas pruebas la completitud y calidad de la aplicación está más cerca.

Para el último tipo de pruebas, las del sistema, se basó su formalización en la experimentación con usuarios. Para estas pruebas se elaboraron un grupo de tareas o actividades que un conjunto de usuarios (un total de doce) iba a ejecutar sobre la aplicación. El grupo de personas que ensayaron con el sistema fue bastante heterogéneo y lo suficientemente grande para poder optimizar el servicio. Estaba compuesto por individuos de diferentes edades, géneros y conocimientos, aunque predominaba la gente joven. Estas pruebas se llevaron a cabo de esta manera, porque se quiere obtener información lo más objetiva posible con sujetos independientes del desarrollo de la aplicación. El conjunto de tareas a realizar era el que se muestra a continuación. Mencionar que no se marcó ningún orden para ejecutar las tareas, aunque es verdad que siguen cierta organización lógica y secuencial. Se pretendía obtener resultados aleatorios de los individuos de prueba.

- Crear un usuario con los datos suministrados en el modo Web.
- Iniciar sesión con el usuario registrado.
- Generar una clave privada propia y guardarla en un fichero de texto.
- Abrir el fichero de la clave privada con la aplicación para introducirlo en el campo correspondiente.
- Insertar la clave privada del usuario para acceder a la aplicación.
- Realizar una búsqueda de eventos.
- Desconectarse de Internet y seguir usando la aplicación.
- Recuperar la conexión a Internet después de realizar algunas operaciones sobre la aplicación.

- Sincronizar los datos del calendario con el servidor.
- Leer un código *QR* de la aplicación con el móvil.
- Navegar entre las opciones de visualización de eventos del calendario.
- Ver y gestionar las notificaciones de los eventos compartidos.
- Aplicar un tema y cambiar el color del texto del calendario.
- Crear mínimo un evento con cualquier información, pero sin añadir asistentes.
- Crear mínimo un evento con cualquier información y con asistentes dentro del formato establecido.
- Crear mínimo un evento con cualquier información en los campos disponibles.
- Cargar eventos mediante una dirección *URL* de la página suministrada.
- Cargar eventos mediante un fichero en formato *iCalendar*.
- Cargar eventos mediante un fichero con el texto cifrado.
- Descargar eventos con el texto plano.
- Descargar eventos con el texto cifrado.
- Visualizar un evento creado.
- Actualizar un evento.
- Eliminar un evento.
- Modificar un evento mediante la interfaz. Arrastrarlo o modificar su forma al final del componente.
- Visualizar eventos que no caben en la casilla del día correspondiente.
- Realizar las operaciones en mínimo una vista diferente.
- Localizar el marcador/elemento del tiempo actual en la aplicación.
- Salir de la aplicación.
- Repetir las actividades, pero ahora en el modo Local.
- Borrar el calendario del modo Local.

- Nota: Apuntar cualquier detalle y/o recomendación a destacar durante la utilización del sistema. Toda información es útil, relevante y de gran ayuda.

Además, estas tareas también se realizaron en diferentes entornos con distintos sistemas operativos, con la petición previa a algunos sujetos, para testear la ejecución de la herramienta *software* y descubrir fallos en la implementación sobre estos contextos. En este ámbito, lo más destacable fue que se encontraron más errores relacionados con la personalización y el diseño, que con la funcionalidad. Los problemas hallados eran cuestiones técnicas vinculadas con la *renderización* y comportamiento del motor de *Chrome* en los sistemas *Ubuntu*. Esto es normal ya que el sistema principal de desarrollo era *Windows*, y era donde se veía y ejecutada la aplicación durante la implementación. Con estas pruebas se pudieron solventar los defectos hallados en estos entornos.

Antes de finalizar, también es importante destacar que se empleó un módulo *JSHint* del *framework Node.js* para verificar la calidad de la codificación de las funciones realizadas. Su funcionamiento es sencillo, pues escanea el código y encuentra posibles problemas de sintaxis, estructurales o estilo según unos parámetros específicos.

Con la aplicación completa de todas y cada una de las actividades y enfoques del conjunto de pruebas se pretende alcanzar el perfeccionamiento del sistema desarrollado. Se sigue este proceso ya que estas tareas se complementan entre sí. Pero crear una aplicación perfecta no es el objetivo, la finalidad es cumplir con los requisitos iniciales que se plantearon para cubrir ciertas necesidades demandadas por los usuarios y alcanzar una calidad del producto *software* alta.

7. Aplicación final

En este apartado se va a exponer el funcionamiento y apariencia del sistema de escritorio multiplataforma desarrollado con el objetivo de suministrar una aplicación capaz de gestionar los eventos de calendario personales de forma segura y ofreciendo privacidad, tanto cuando se utilice con conexión a Internet como un servicio web como cuando se ejecute en modo local sin la necesidad de emplear recursos de red. Además, el sistema final soporta la gestión de pérdida de conexión cuando se está trabajando en modo Web.

A continuación, se van a mostrar algunas pantallas de la interfaz de la aplicación. Las pantallas representadas van a ser indistintamente de un entorno *Windows* o *Ubuntu*, para así poder visualizar la interfaz en diferentes sistemas operativos de ejecución. El sistema es fácilmente portable entre las diversas plataformas, debido a que el *framework Electron* ofrece unos componentes y características útiles para realizar el empaquetado de la herramienta *software*, generando una aplicación adecuada para la utilización en los distintos contextos.

Destacar que las imágenes mostradas son de la aplicación funcionando con un formato horizontal a una resolución *HD* de 1280x720 pixeles, la cual es la mínima resolución a la que funciona el sistema. Las resoluciones mayores son soportadas, adaptando la interfaz a las nuevas especificaciones de pantalla del usuario.

Las pantallas expuestas son los estados más importantes que se pueden encontrar en el sistema. Cada una de las imágenes a visualizar tiene una explicación de las funcionalidades que se suministran y detalles de los componentes que en ellas se hallan. Además, en el fondo de las pantallas se pueden diferenciar las diferentes vistas de representación del calendario. Éstas son: agenda semanal, agenda diaria, mensual, semanal y diaria.

- Pantalla de inicio

En esta pantalla (Ver Figura 17) se puede apreciar el diálogo con los elementos encargados de mostrar el estado de la conexión de red, seleccionar el modo de utilización de la aplicación, suministrar formas de acceso, y el formulario para obtener los datos del usuario. Si la autenticación se realiza de forma correcta se navega a la pantalla de clave del usuario.

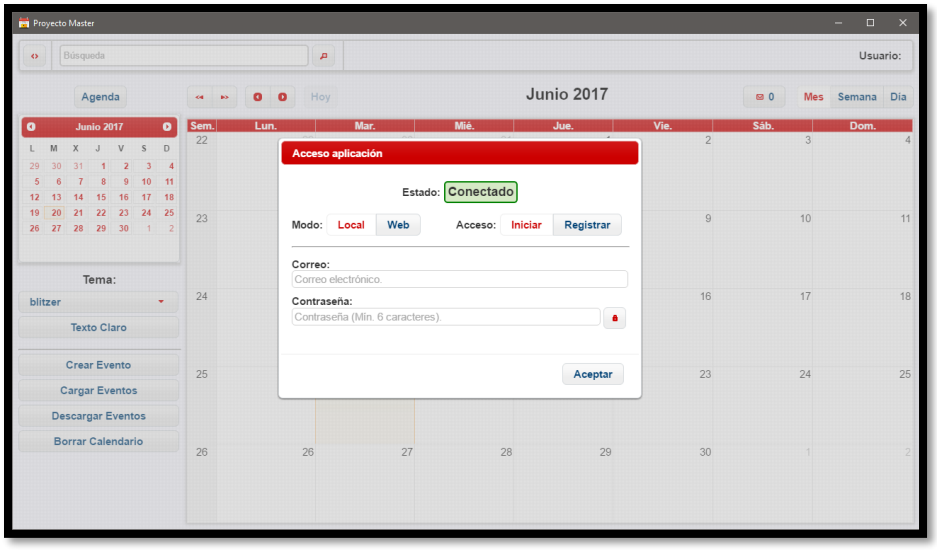

Figura 17: Interfaz de inicio

- Pantalla de clave de usuario

En esta pantalla (Ver Figura 18) se muestra el diálogo para gestionar la clave privada del usuario. En ella se aprecia la información del usuario que ha iniciado sesión, el formulario para obtener los datos de la clave privada, y las funciones para abrir un fichero con la clave privada, crear una clave privada con contraseña, insertar una clave privada en la aplicación y salir del sistema. Además, también se puede observar un ejemplo de clave privada. Si la validación de la clave privada es afirmativa se navega a la pantalla del calendario.

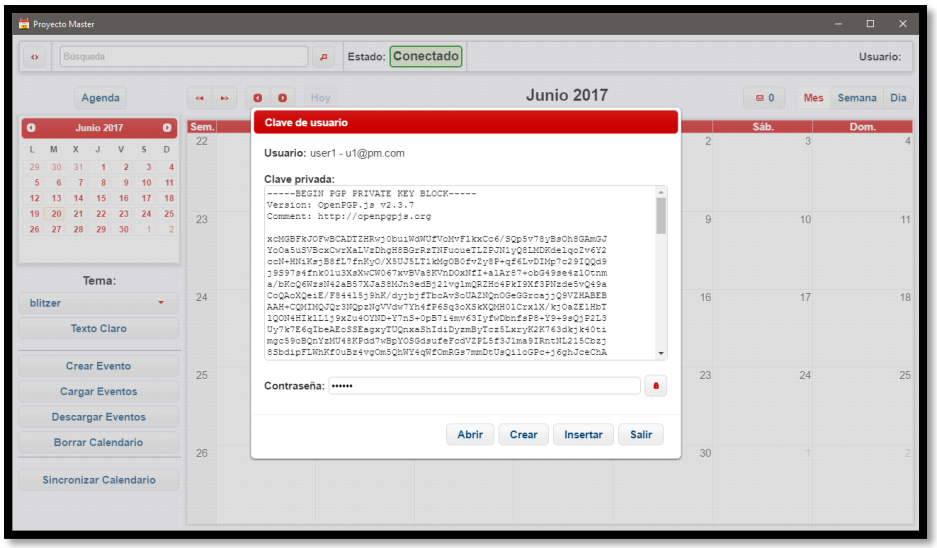

Figura 18: Interfaz de clave de usuario

- Pantalla del calendario

En esta pantalla (Ver Figura 19) se observa el estado principal de la aplicación. En ella se pueden ver elementos que muestran información del sistema, representan los eventos del calendario, permiten realizar búsquedas de eventos, manifiestan el estado de la conexión, gestionan la información del usuario, suministran la navegación por el calendario, administran las notificaciones de eventos compartidos, manejan la personalización del calendario, gestionan los eventos del calendario y posibilitan la salida del sistema. Las funcionalidades de esta pantalla se describen detalladamente en pantallas posteriores.

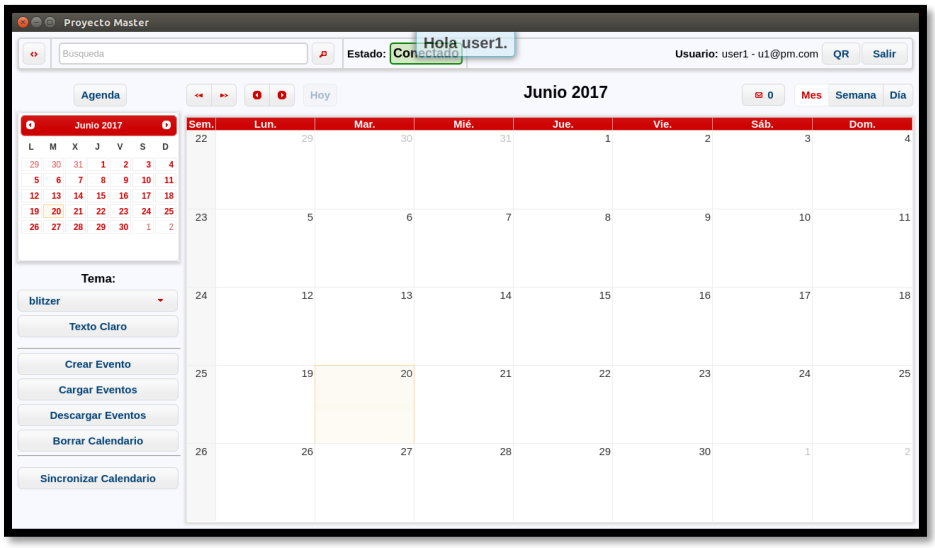

Figura 19: Interfaz del calendario

- Pantalla para crear eventos

En esta pantalla (Ver Figura 20) se muestra el diálogo con el formulario a rellenar para poder añadir un evento al calendario. Los campos del evento son: el título, el color, el indicador de diario u horario, el indicador de disponible u ocupado, la fecha y hora de inicio, la fecha y hora de fin, el lugar, la *URL*, la descripción y los asistentes. El organizador del evento es el usuario con la sesión activa en la aplicación. Las funciones disponibles son la confirmación de la creación del evento o su correspondiente cancelación. La navegación de las dos acciones es a la pantalla del calendario.

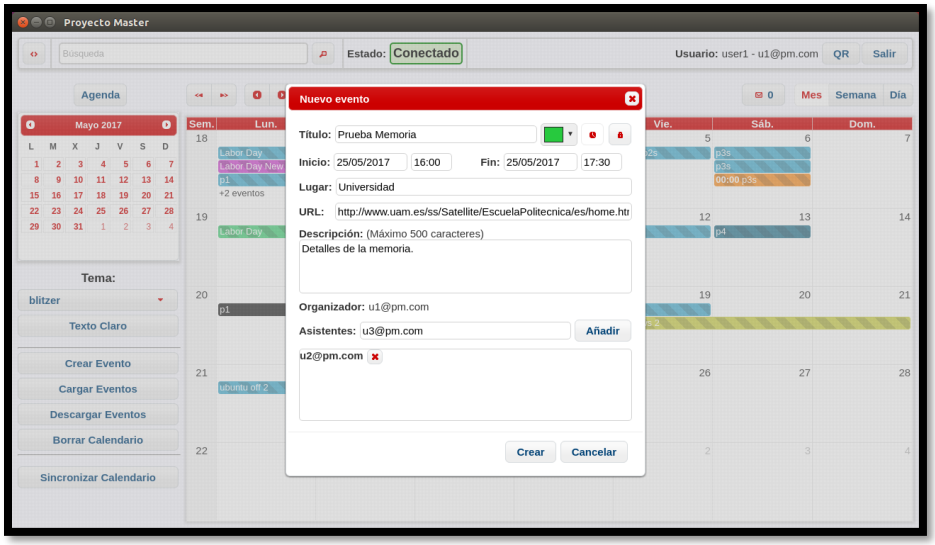

Figura 20: Interfaz para crear eventos

- Pantalla para cargar eventos

En esta pantalla (Ver Figura 21) se puede ver el diálogo con el formulario para insertar eventos externos de forma masiva. Los componentes representados permiten seleccionar el modo de carga y el color de los elementos a insertar. En función del modo, se habilitan las opciones para seleccionar un fichero o introducir una dirección *URL*. Las funciones disponibles son la adición de los eventos o la cancelación del proceso. Ambas acciones al finalizar realizan la navegación a la pantalla del calendario.

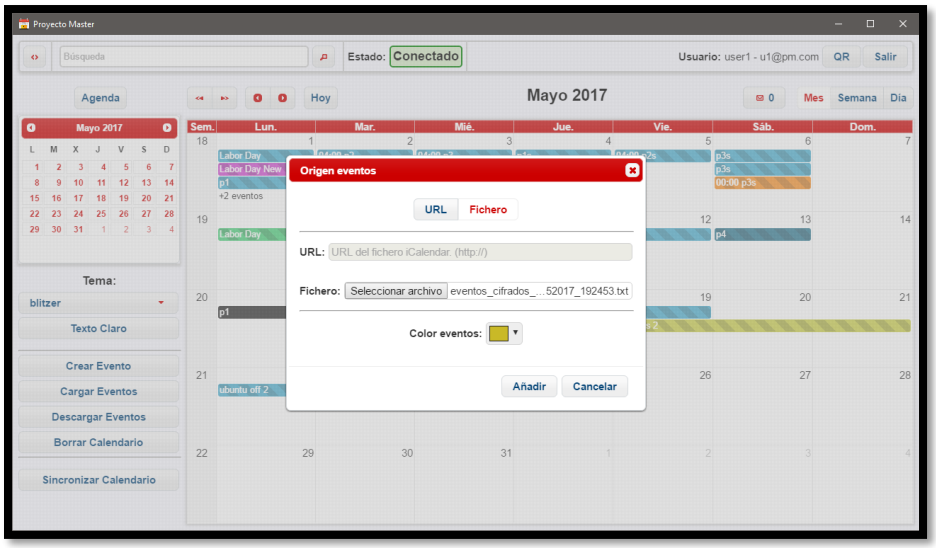

Figura 21: Interfaz para cargar eventos

- Pantalla para descargar eventos

En esta pantalla (Ver Figura 22) se observa el diálogo con las funciones para seleccionar el formato del contenido del fichero a descargar con los eventos del calendario. Al elegir el formato de texto cifrado los eventos descargados son cifrados con la clave pública del usuario de la sesión. Al marcar el formato texto plano los eventos son descargados en un fichero con la estructura *iCalendar*, por lo que los elementos son legibles. Hay una acción disponible, cancelar, con la que se navega a la pantalla del calendario.

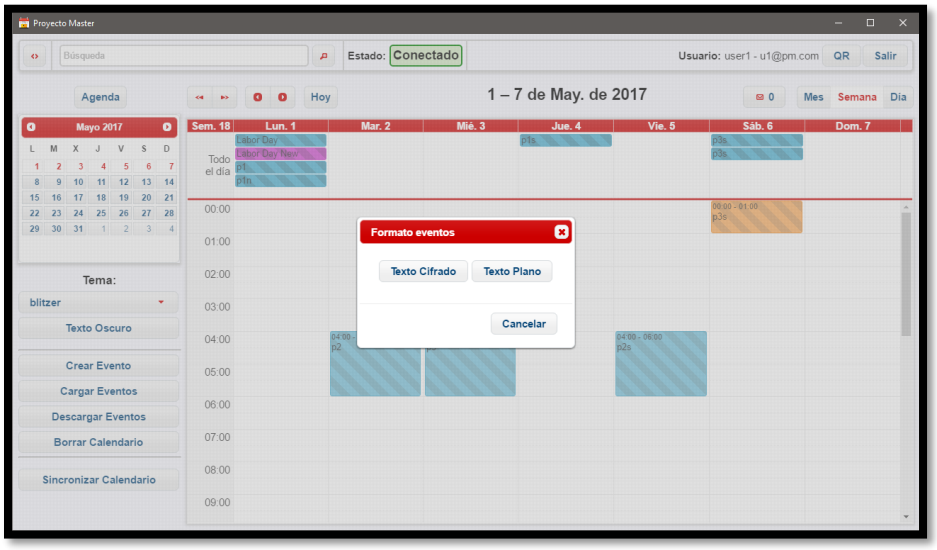

Figura 22: Interfaz para descargar eventos

- Pantalla de búsqueda y notificaciones de eventos compartidos

En esta pantalla (Ver Figura 23) se exponen los cuadros de diálogo obtenidos cuando se realiza una búsqueda (parte superior izquierda) y cuando se gestionan las notificaciones de eventos compartidos (parte superior derecha). Al efectuar la primera acción, se muestran los eventos que contienen en su título el criterio de búsqueda ingresado. Con la segunda acción, se puede contemplar el estado del proceso de compartición de eventos con los asistentes. Los estados corresponden con la siguiente definición: los mensajes en color rojo significan que el evento no se pudo compartir con ninguno de los asistentes debido a problemas de conexión, los mensajes en color verde expresan que el evento se pudo compartir con la lista de asistentes que contienen en su interior, y los mensajes en color amarillo informan que el evento no se pudo compartir con la lista de asistentes que figuran en su encabezado.

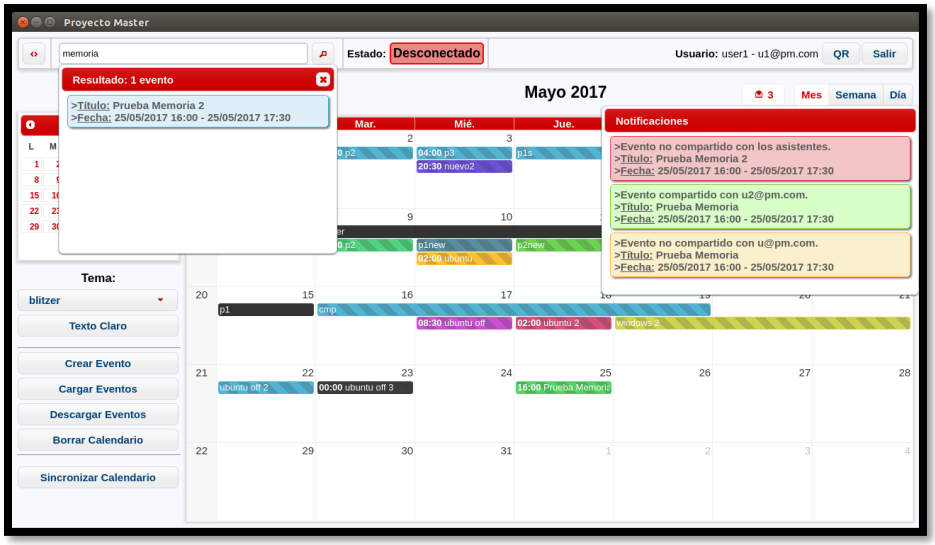

Figura 23: Interfaz de búsqueda y notificaciones de eventos compartidos

- Pantalla de personalización y visualización del calendario

En esta pantalla (Ver Figura 24) se expone una nueva apariencia de la aplicación después de haber seleccionado un tema distinto al que se muestra por defecto. La elección del tema se realiza mediante la función de tema del calendario (parte central izquierda). También, se puede observar como el calendario expone los eventos con una vista distinta. En este caso se representa la vista de agenda semanal.

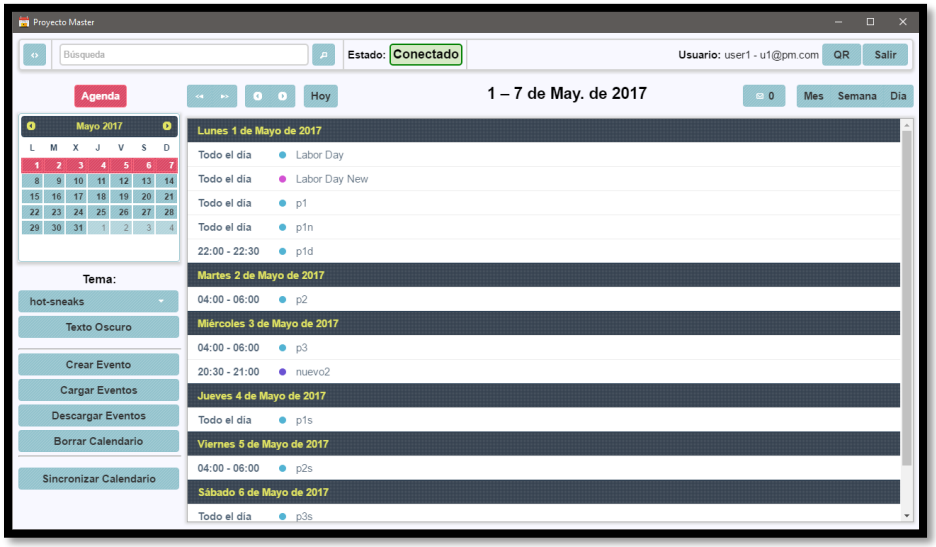

Figura 24: Interfaz de personalización y visualización del calendario

- Pantalla de información de eventos

En esta pantalla (Ver Figura 25) se puede visualizar la información específica de un evento seleccionado del calendario del usuario. Las funciones disponibles son obtener un código *QR* con la información del evento, eliminar el evento, actualizar el evento y cancelar la visualización. Al elegir la acción *QR* se muestra en la pantalla, de manera superpuesta con la información del evento, el código *QR* generado. Al seleccionar la acción eliminar o cancelar se navega a la pantalla del calendario. Al marcar la acción actualizar se navega a la pantalla para actualizar eventos, la cual es semejante a la usada para crear eventos, sino que en esta ocasión el formulario aparecería relleno con los datos del evento que se había seleccionado.

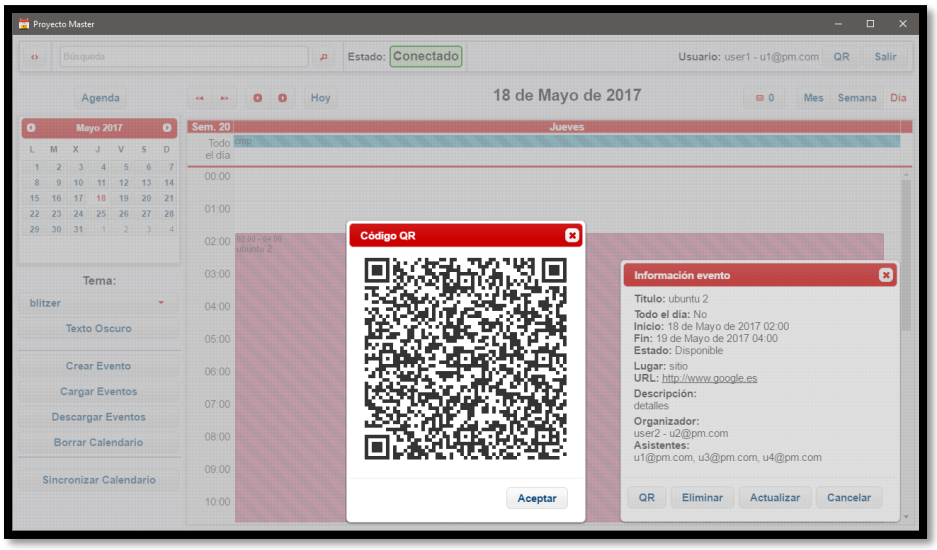

Figura 25: Interfaz de información de eventos

- Pantalla de código *QR*

En esta pantalla (Ver Figura 26) se ve el diálogo que muestra la representación de la clave del usuario mediante un código *QR*. Para realizar esta acción es necesario elegir la función *QR* de las opciones de usuario (parte superior derecha). Este recurso puede ser decodificado mediante una aplicación adecuada en un dispositivo móvil. La acción disponible, aceptar, navega a la pantalla del calendario.

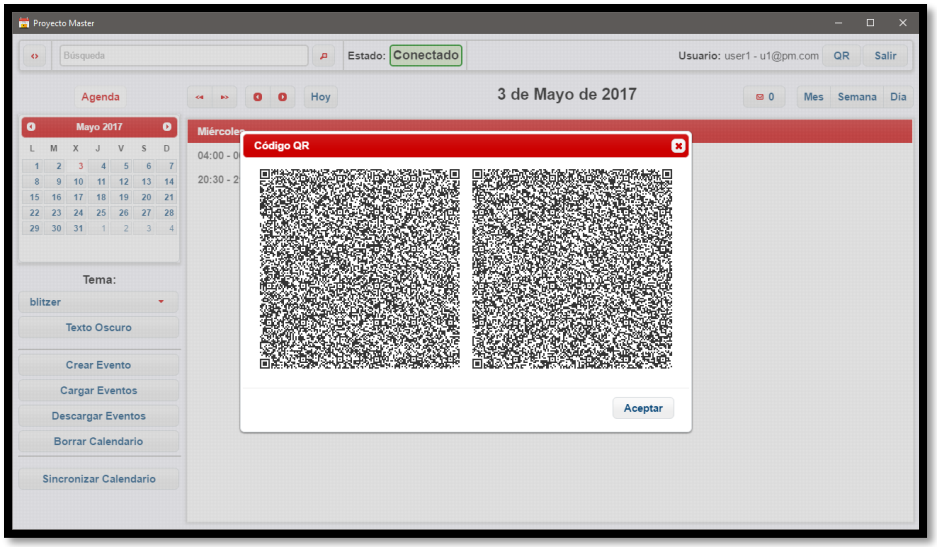

Figura 26: Interfaz de código QR

- Pantalla de sincronización de calendario y notificaciones de escritorio

En esta pantalla (Ver Figura 27) se muestra el mensaje del estado de la sincronización cuando se ejecuta dicha función seleccionando la acción sincronizar calendario (parte inferior izquierda). Además, se puede ver una notificación de escritorio (parte superior derecha) con la información de que se ha compartido un evento pendiente. Esto sucede cuando se comparten eventos sin conexión, pues al estar en dicha situación, los eventos no se han podido compartir, pero cuando se recupera la conexión y se realiza la acción de sincronizar calendario, los eventos que estaban pendientes de compartir, se envían a sus asistentes.

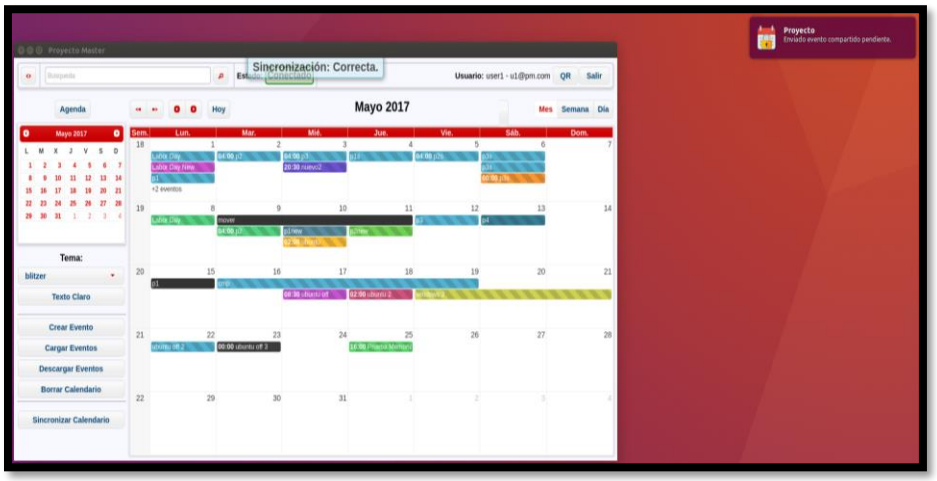

Figura 27: Interfaz de sincronización de calendario y notificaciones de escritorio

- Pantalla de pérdida de conexión

En esta pantalla (Ver Figura 28) se puede observar el diálogo con el mensaje que aparece cuando durante la utilización de la aplicación en el modo Web se pierde la conexión. Asimismo, en el cuadro con el estado de la conexión del calendario principal se visualiza que la conexión a Internet se ha suspendido.

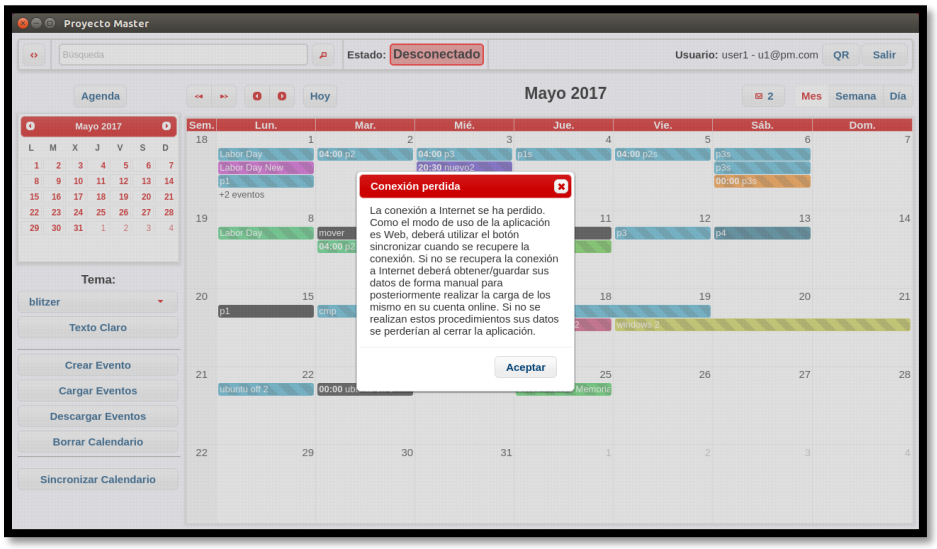

Figura 28: Interfaz de pérdida de conexión

- Pantalla de notificaciones de carga de eventos compartidos

En esta pantalla (Ver Figura 29) se expone el cuadro de diálogo de gestión de notificaciones de carga de eventos compartidos (parte superior derecha). En el interior del administrador de notificaciones se pueden encontrar diferentes tipos de estados de mensajes: los mensajes en color azul significan que un evento compartido ha sido creado en la aplicación, los mensajes en color rojo describen que un evento compartido ha sido eliminado del sistema, y los mensajes en color amarillo representan que un evento compartido ha sido actualizado en el calendario. Todos los mensajes contienen mínima información del evento cargado. Además, también se puede ver una notificación de escritorio (parte inferior derecha) que informa de la carga de eventos compartidos en el calendario personal. Igualmente, se puede apreciar que el calendario principal se muestra en toda la pantalla. Esto es debido a que se ha seleccionado la función de contraer la barra lateral de la aplicación (parte superior izquierda).

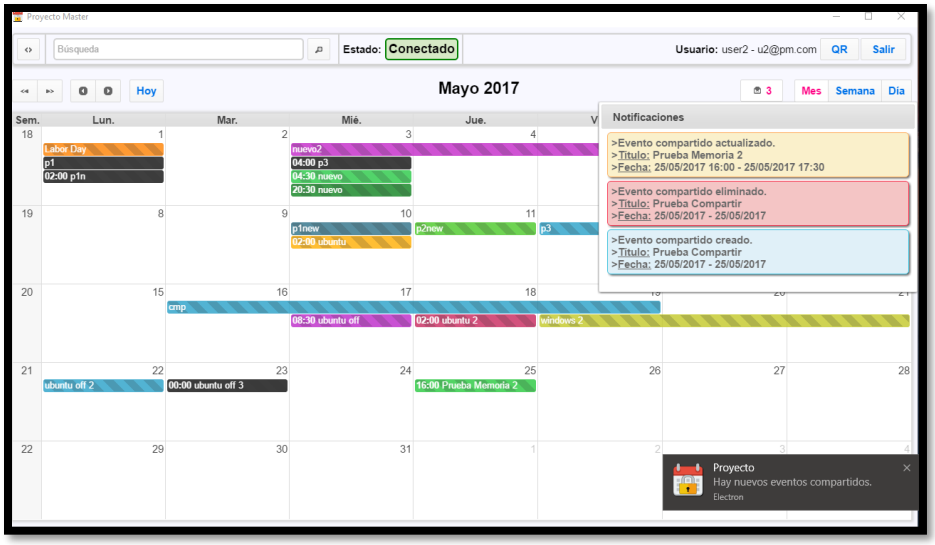

Figura 29: Interfaz de notificaciones de carga de eventos compartidos

- Pantalla para borrar calendario

En esta pantalla (Ver Figura 30) se muestra el diálogo con un mensaje informativo cuando se quieren borrar todos los eventos personales mediante la selección de la función borrar calendario (parte inferior izquierda). Hay disponibles dos acciones, una es sí, que confirmaría la eliminación de los datos, y la otra es no, que cancelaría el proceso de borrado. Con ambas funciones se navega a la pantalla del calendario.

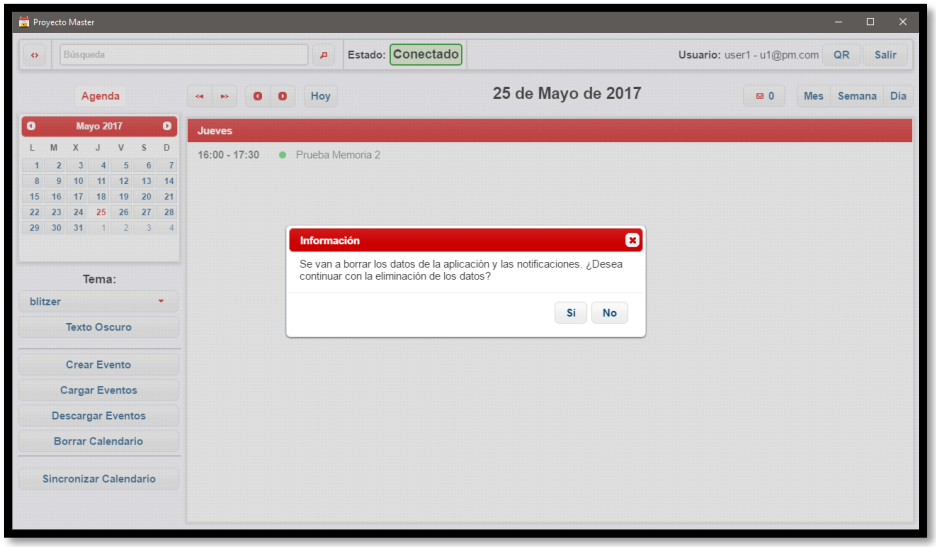

Figura 30: Interfaz para borrar calendario

8. Conclusiones y trabajo futuro

Después de realizar todo el trabajo de desarrollo vinculado al proyecto se van a exponer las ideas y conclusiones que se han obtenido con su elaboración. También, se van a mencionar vías y cuestiones de trabajo futuro que serían convenientes abordar para mejorar y ampliar las funcionalidades de la aplicación de escritorio multiplataforma creada, con la intención de dotar de completitud al entorno, y así ofrecer una mejor experiencia a los usuarios.

8.1. Conclusiones

El presente trabajo de fin de máster surgió de las ideas concebidas y expuestas en el proyecto inicial realizado en torno al trabajo fin de grado de ingeniería informática [1] hecho en esta facultad, la Escuela Politécnica Superior (Universidad Autónoma de Madrid). Ese trabajo sirvió de punto de partida para elaborar los conceptos desarrollados en la nueva aplicación. Además, ayudó a sentar las bases del conocimiento y a obtener la experiencia requerida para abordar la creación de este tipo de proyectos tecnológicos, en todas sus etapas. Igualmente, es importante incidir que con la elaboración de este trabajo se pretenden aplicar los conocimientos y habilidades adquiridos durante la formación en el máster.

Tras la asunción de esos aspectos, una de las cuestiones relevantes en el recorrido del proyecto fue la creación de un entorno nativo que pudiera ejecutarse en diferentes sistemas operativos. Gracias a las tecnologías web y al *framework Electron*, este objetivo se pudo conseguir. Además, este enfoque de desarrollo permitió abordar la necesidad de suministrar una aplicación independiente de los recursos de red.

Aunque las tecnologías web en la actualidad son bastante potentes, aún les quedan aspectos por cubrir en la implementación del lado del cliente [109]. Pero viendo su rápida evolución, no queda duda que serán herramientas altamente demandadas y usadas en un futuro cercano, pues ya serán

capaces de ofrecer en el entorno local los medios necesarios para realizar las tareas que se han tenido que llevar al lado del servidor. Con ese progreso se conseguirá solventar cualquier problema o imposibilidad técnica que se pueda tener ahora.

Durante las fases de creación de la herramienta *software* se optó por el uso de un enfoque basado en que los recursos del lado del servidor se obtenían como servicios (*Backend as a Service, BaaS*). Para ello, se utilizó la herramienta *Firebase* que cumplió con los requerimientos de la aplicación. Esta perspectiva de manejo del servidor, sin que el desarrollador de las aplicaciones sea el administrador y/o el propietario del mismo, hace que se cree cierta relación de dependencia con elementos externos, lo que puede constituir un riesgo en el ciclo de vida del producto *software*, si dicha dependencia no es debidamente tratada. Este nuevo contexto de desarrollo se emplea cada vez con mayor frecuencia, y la adopción de un entorno *sin servidor* suministra nuevas maneras de orientar y crear las aplicaciones (arquitectura *Serverless* [110]).

Además, con el propósito de reducir la dependencia adquirida con los servicios *cloud*, y teniendo presente que el sistema está fundamentado en el patrón de diseño *SPA*, se hizo necesario seguir metodologías y principios básicos que apoyaran la implementación del servicio. En este sentido, se examinaron las buenas prácticas recomendadas por la iniciativa *The Twelve-Factor App* [111], que establece doce principios que deberían incorporar las aplicaciones que van a ser ofrecidas como servicios.

Por otro lado, en el ámbito de la seguridad y la privacidad, se ha querido crear un sistema que sea respetuoso con la información de los usuarios, y que suministre e incorpore los elementos idóneos para asegurar los contenidos. Estas características han sido fundamentales y muy consideradas durante el desarrollo de la aplicación. La opción adoptada en el proyecto está en la línea de lo demandado por la nueva legislación europea sobre la protección de datos (*General Data Protection Regulation, GDPR* [112]), que exige proteger tanto la seguridad de las aplicaciones y servicios, como la privacidad de los usuarios. Además, junto con estos elementos fue considerada la usabilidad del entorno, pues de casi nada sirve que un sistema sea seguro y garantice el principio del consentimiento informado [113], si los propietarios de los datos, que son los que van a manejar la aplicación, no son capaces de poder gestionar su información con facilidad.

Como conclusión final, la creación del sistema de escritorio multiplataforma para la gestión de eventos de calendario de modo seguro y confidencial culminó con éxito, de modo que la aplicación desarrollada satisface los requisitos que se habían concretado en la fase de análisis.

8.2. Trabajo futuro

Como ya se había comentado en apartados anteriores, la aplicación no se concibe con la idea de ser perfecta, por lo que aún se podrían añadir nuevas características o incluir nuevos enfoques y vías de desarrollo. Algunas de las ideas que se podrían tratar como trabajo futuro son:

- La adición de nuevos y diferentes mecanismos de seguridad, como la utilización de certificados digitales o de *tokens hardware* para el manejo de las claves asimétricas [114]. Estas mejoras permiten separar el canal de datos del canal de gestión de identidades, lo que dificultaría la inclusión de identidades falsas por parte del proveedor de servicio, es decir, *Firebase*.

- La inclusión de los otros procedimientos de autenticación de usuarios que proporciona el servicio *Firebase*. En efecto, *Firebase* proporciona mecanismos de autenticación anónimos que pueden ser de interés para la gestión de determinados eventos de calendario.

- La incorporación de nuevas funcionalidades, tanto para el sistema como para el calendario. Algunos nuevos componentes a estudiar podrían ser una herramienta de chat para comunicar a los usuarios de la aplicación, o añadir nuevos elementos en los eventos de la interfaz, como los grupos de usuarios.

- La interacción con otros dispositivos o crear un entorno que pueda ser ejecutado en los sistemas de los dispositivos móviles. Se podrían analizar las formas, a poder ser relacionadas con las tecnologías web, para extender el núcleo del proyecto, y así, intentar abarcar todo el conjunto de dispositivos del mercado.

- La agregación de otros métodos de compartición de archivos que continúen con el enfoque de la seguridad como tema predominante [115].

- La seguridad de cualquier herramienta *software* debe ser evaluada y validada de modo exhaustivo. En este proyecto la seguridad y la privacidad han sido considerados en cada una de las fases de desarrollo. No obstante, quedaría efectuar un análisis formal del protocolo de gestión de eventos de acuerdo, por ejemplo, con la metodología reseñada en [116].

- La usabilidad del actual prototipo desarrollado en este TFM ha sido evaluada en una primera aproximación. Como parte de la validación final de la herramienta, habría que considerar los principios de Nielsen [117]. Además, el usuario es una pieza fundamental en el manejo seguro y eficaz de la herramienta. Por ello, sería recomendable tener en consideración recientes propuestas que incluyen al usuario como parte esencial en la validación de protocolos de seguridad [118].

Glosario

- Algoritmo de cifrado: Conjunto de operaciones criptográficas que permiten cifrar y descifrar textos.
- Antivirus: Programa que detecta la presencia de virus en un sistema y puede realizar acciones para contener sus efectos.
- API (*Application Programming Interface*): La interfaz de programación de aplicaciones es un conjunto de métodos y procesos que suministra una biblioteca para ser usada por otro elemento. Es una capa de abstracción, interfaces de las funciones.
- Aplicación nativa: Es la herramienta que está desarrollada y optimizada para un sistema operativo concreto.
- Backend: Es la capa de acceso a los datos. Son los recursos que están en el lado del servidor.
- Backup: Copia de seguridad de la información original fuera de la infraestructura principal con el propósito de tener una forma de recuperar los datos en caso de pérdida.
- Balanceador de carga: Dispositivo que se encuentra delante de un grupo de servidores que ofrecen una aplicación y con el fin de asignar las solicitudes de los clientes usando un algoritmo de reparto.
- Ciberataque: Ataque que se realiza a través de Internet.
- Ciberguerra: Conflicto que tiene como campo de combate la red y las tecnologías de la información.
- Cifrar: Procedimiento que emplea un algoritmo de cifrado con una clave para ocultar un mensaje.
- Clave de cifrado: Elemento de información que controla el proceso de un algoritmo criptográfico.
- Cloud: Referencia a Internet sobre los recursos que se encuentran en la nube.
- Cloud computing: La computación en la nube es un paradigma que ofrece capacidad de cálculo y procesamiento a través de Internet.
- Cloud storage: El almacenamiento en la nube es un paradigma que suministra recursos de almacenamiento en Internet.

- Confidencialidad: Propiedad de la información con la que se garantiza que sólo es accesible a elementos autorizados.
- Criptografía: Conjunto de técnicas de cifrado cuyo objetivo es ocultar la información mediante la modificación de la representación de los textos.
- CSS (*Cascading StyleSheets*): Las hojas de estilo en cascada es un lenguaje de diseño gráfico para especificar y crear la presentación de un fichero estructurado escrito con un lenguaje marcado.
- DDoS (*Distributed Denial of Service*): Un ataque de denegación de servicio distribuido se basa en atacar los servidores de un recurso desde distintos orígenes con la intención de impedir el acceso a los usuarios finales.
- Decodificar: Proceso que aplica un conjunto de reglas sobre un mensaje para conseguir su información.
- Descifrar: Procedimiento que emplea un algoritmo de cifrado con una clave para obtener la información de un mensaje cifrado.
- Empaquetar: Proceso para suministrar las aplicaciones o programas en forma de paquetes. Son los ejecutables de la aplicación.
- Explotar: Proceso de aprovechar una vulnerabilidad de seguridad en un sistema y obtener un comportamiento anómalo.
- Firewall: Un cortafuegos es un dispositivo o sistema que se encarga de administrar los accesos de red, bloqueando los no autorizados y permitiendo los contrarios.
- Framework: Un marco de trabajo es un conjunto de conceptos, criterios y tecnologías de ayuda específica que sirven de apoyo para la estructuración y desarrollo de aplicaciones.
- Gusano: Es un programa malicioso que tiene la propiedad de expandirse y reproducirse.
- Hackear: Proceso de acceder sin autorización a un sistema o red.
- Hacktivismo: Acrónimo de hacker y activismo, que representa el uso no violento de las herramientas informáticas con fines políticos.
- Hardware: Son los componentes físicos de un sistema informático.
- HTML (*HyperText Markup Language*): El lenguaje de marcas de hipertexto es un lenguaje para la elaboración de páginas web.

- IaaS (*Infrastructure as a Service*): La infraestructura como servicio es una forma de suministrar recursos de almacenamiento básicos y capacidades de cálculo como servicios en la red.
- IDS (*Intrusion Detection System*): Un sistema de detección de intrusiones es un dispositivo o programa que encuentra accesos no autorizados en un entorno informático.
- Infraestructura: Conjunto de medios técnicos, servicios e instalaciones necesarios para el desarrollo y cumplimiento de una actividad, o para que un lugar pueda ser usado.
- Intrusión: Acción de acceso de un elemento de manera indebida.
- Inyección de código SQL (*Structured Query Language*): Es una técnica de introducción de código malicioso que se basa en una vulnerabilidad relacionada con las bases de datos.
- IPS (*Intrusion Prevention System*): Un sistema de prevención de intrusiones es un dispositivo o programa que controla los accesos en un entorno informático para asegurar los sistemas de acciones maliciosas.
- JavaScript: Es un lenguaje de programación interpretado, débilmente tipado y dinámico. Se utiliza para ofrecer mejoras en las interfaces de usuario y contenidos dinámicos en las páginas web.
- JSON (*JavaScript Object Notation*): Es un formato de texto ligero o estructura para el intercambio de datos.
- Lado del cliente (*Client-Side*): Es el contexto donde el usuario realiza sus operaciones en un entorno informático.
- Lado del servidor (*Server-Side*): Es el contexto donde se procesan las operaciones y peticiones del usuario en un entorno informático.
- Librería: Una biblioteca es un conjunto de implementaciones codificadas mediante un lenguaje de programación, que suministra unas interfaces específicas para una funcionalidad.
- Malicioso: Cualidad que representa intenciones encubiertas para beneficiarse de algo o perjudicar algo.
- Maquetar: Estructurar u organizar los elementos de una página web.
- Modularidad: Capacidad que tiene un sistema de ser organizado o estructurado en varias partes que se relacionan con un propósito común.

- Multiplataforma: Cualidad que representa la utilización en diferentes entornos o sistemas operativos.
- Offline: Estado de sin conexión o desconectado de Internet.
- Online: Estado de en línea, conectado o con conexión a Internet.
- OpenSource: El código abierto o libre es aquel que tiene los derechos de propiedad bajo dominio público.
- PaaS (*Platform as a Service*): La plataforma como servicio es la abstracción de un contexto de desarrollo y empaquetamiento de un conjunto de recursos que ofrecen una funcionalidad horizontal como servicios.
- Phishing: La suplantación de identidad es un concepto de abuso informático cuyo objetivo es conseguir información confidencial de modo fraudulento.
- Plugin: Un complemento es una aplicación que interacciona con otra para añadirle nuevas funcionalidades concretas.
- Privacidad: Propiedad de la información con la que se garantiza que sólo es accesible por el propietario/dueño.
- Proxy: Es un dispositivo o programa que actúa de intermediario en las peticiones de los recursos de los usuarios.
- Ransomware: Es una aplicación maliciosa que restringe el acceso a los recursos del sistema infectado y que pide un rescate para suprimir el bloqueo.
- Renderizar: Es el proceso de generación de imágenes.
- SaaS (*Software as a Service*): El *software* como servicio es el suministro de aplicaciones completas como servicios.
- Sistema operativo: Es el *software* principal o conjunto de programas de un sistema que se encarga de administrar los elementos *hardware* y de suministrar servicios a las aplicaciones.
- Software: Es el conjunto lógico de un sistema informático que ofrece la posibilidad de realizar tareas concretas.
- Spam: Es el correo electrónico basura no deseado que se envía a un gran conjunto de destinatarios con intención publicitaria o comercial.
- Texto cifrado: Tipo de texto no legible y que es el resultado de aplicar métodos criptográficos mediante un algoritmo y una clave.

- Texto plano: Tipo de texto que es legible, pues aún no se han aplicado métodos criptográficos, o ya se han empleado, pero en modo descifrado.
- Token: Es un elemento de seguridad que permite facilitar el proceso de autenticación de usuarios autorizados a ciertos entornos informáticos. Estos dispositivos guardan claves criptográficas.
- Ubicuo: Cualidad que representa el acceso desde distintas partes o medios.
- Virtualizar: Es la creación de una versión virtual de un recurso tecnológico específico mediante aplicaciones y herramientas destinadas a dicho fin.
- Virus: Es un programa malicioso que pretende modificar el funcionamiento del sistema infectado sin la autorización o el conocimiento del usuario propietario.
- VPN (*Virtual Private Network*): Una red privada virtual es una tecnología de red que posibilita la extensión segura de la red de área local usando recursos públicos de comunicación.
- Vulnerabilidad: Es una debilidad de un entorno informático que puede ser utilizada para causar algún daño o perjuicio.
- Wearable: Es el concepto que se aplica a la tecnología vestible o complementos inteligentes.
- XSS (*Cross-Site Scripting*): Es un tipo de vulnerabilidad informática típica de las aplicaciones web. Su objetivo es permitir que agentes maliciosos puedan inyectar en estos entornos visitados por los usuarios un tipo de código o lenguaje concreto para eludir los mecanismos de control.

Referencias

[1] R. X. Pico Paredes, "Diseño de un sistema de gestión segura de eventos de calendario en la nube," *Trabajo Fin de Grado, Escuela Politécnica Superior, Universidad Autónoma de Madrid*, 2015. [Online]. Available: https://repositorio.uam.es/handle/10486/668426.

[2] "EU Cybersecurity plan to protect open internet and online freedom and opportunity - Cyber Security strategy and Proposal for a Directive," Feb-2013. [Online]. Available: http://ec.europa.eu/digital-agenda/en/news/eu-cybersecurity-plan-protect-open-internet-and-online-freedom-and-opportunity-cyber-security. [Accessed: 27-Jun-2017].

[3] Silicon, "PayPal Phishing Attack Demands Selfie With Photo ID." [Online]. Available: http://www.silicon.co.uk/security/paypal-phishing-selfiie-photo-215111?inf_by=594e3. [Accessed: 31-May-2017].

[4] T. Register, "Identity management outfit OneLogin sugar coats impact of attack." [Online]. Available: https://www.theregister.co.uk/2017/06/01/onelogin_breached/. [Accessed: 31-May-2017].

[5] Enisa, "Online privacy tools for the general public." [Online]. Available: https://www.enisa.europa.eu/publications/privacy-tools-for-the-general-public/at_download/fullReport. [Accessed: 31-May-2017].

[6] J. Hoepman and B. Jacobs, "Software Security Through Open Source," pp. 1–15, 2005.

[7] T. W. S. Journal, "All IT Jobs Are Cybersecurity Jobs Now." [Online]. Available: https://www.wsj.com/articles/all-it-jobs-are-cybersecurity-jobs-now-1495364418. [Accessed: 31-May-2017].

[8] Wikipedia, "Information technology." [Online]. Available: https://en.wikipedia.org/wiki/Information_technology. [Accessed: 10-May-2017].

[9] H. J. Leavitt and T. L. Whisler, *Management in the 1980's*. November, 1958.

[10] T. K. Derry and T. I. Williams, *Historia de la tecnología*, vol. 1. Siglo XXI de España Editores, 1977.

[11] G. Lin, D. Fu, J. Zhu, and G. Dasmalchi, "Cloud computing: IT as a service," *IT Prof. Mag.*, vol. 11, no. 2, p. 10, 2009.

[12] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[13] J. McCarthy, "Reminiscences on the history of time sharing," *URL http//www-formal. stanford. edu/jmc/history/timesharing. dvi*, 1983.

[14] S. Garfinkel and H. Abelson, *Architects of the information society: 35 years of the Laboratory for Computer Science at MIT*. MIT press, 1999.

[15] B. M. Leiner *et al.*, "A brief history of the Internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 5, pp. 22–31, 2009.

[16] Wikipedia, "History of the Internet." [Online]. Available: https://en.wikipedia.org/wiki/History_of_the_Internet. [Accessed: 12-May-2017].

[17] M. Böhm, S. Leimeister, C. Riedl, and H. Krcmar, "Cloud computing and computing evolution," *B. Smith,"An approach to graphs linear forms (Unpublished Work style)," Unpubl.*, 2010.

[18] A. G. Molina, "Las TICs y la Ley de Moore." [Online]. Available: https://telos.fundaciontelefonica.com/url-direct/pdf-generator?tipoContenido=articulo&idContenido=2010011910550001. [Accessed: 24-Jun-2017].

[19] Wikipedia, "Cloud computing." [Online]. Available: https://en.wikipedia.org/wiki/Cloud_computing. [Accessed: 15-May-2017].

[20] Wikipedia, "Chrome OS." [Online]. Available: https://en.wikipedia.org/wiki/Chrome_OS#cite_ref-9. [Accessed: 20-May-2017].

[21] Hipertextual, "Google: 'Chrome OS está aquí para quedarse.'" [Online]. Available: https://hipertextual.com/2015/11/chrome-os-quedarse. [Accessed: 20-May-2017].

[22] F. Webster, *The information society revisited*. Sage, retrieved from http://books. google. co. uk/books, 2005.

[23] M. Peñarroya, “Evolución de la Sociedad de la Información en España.” [Online]. Available: http://www.montsepenarroya.com/evolucion-de-la-sociedad-de-la-informacion-en-espana/. [Accessed: 20-May-2017].

[24] D. C. J. de Parga, *Cloud computing: retos y oportunidades*. Fundaci{ó}n Ideas, 2011.

[25] L. J. Aguilar, “La Computación en Nube (Cloud Computing): El nuevo paradigma tecnológico para empresas y organizaciones en la Sociedad del Conocimiento,” *Rev. Icade. Rev. las Fac. Derecho y Ciencias Econ{ó}micas y Empres.*, no. 76, pp. 95–111, 2012.

[26] BBVA, “Ocho razones para usar el ‘cloud computing’ en la empresa.” [Online]. Available: https://www.bbva.com/es/ocho-razones-para-usar-el-cloud-computing-en-la-empresa/. [Accessed: 21-May-2017].

[27] ElFuturoEsOne, “El aula multipantalla que no necesita pupitres.” [Online]. Available: https://www.youtube.com/watch?v=G8GVBdkb-v8. [Accessed: 20-May-2017].

[28] Genbeta, “Cloud computing, un futuro muy prometedor para el empleo en el sector TIC.” [Online]. Available: https://www.genbeta.com/n/cloud-computing-un-futuro-muy-prometedor-para-el-empleo-en-el-sector-tic. [Accessed: 21-May-2017].

[29] LibertadDigital, “El futuro de la educación está en enseñar a programar en la escuela El futuro de la educación está en enseñar a programar en la escuela.” [Online]. Available: http://www.libertaddigital.com/ciencia-tecnologia/tecnologia/2015-05-09/el-futuro-de-la-educacion-esta-en-ensenar-a-programar-en-la-escuela-1276547574/. [Accessed: 21-May-2017].

[30] elFuturoEsOne, “La universidad sin profesores ni exámenes donde estudian los futuros genios de la programación.” [Online]. Available: https://www.youtube.com/watch?v=Am9Q7I6JkiE. [Accessed: 21-May-2017].

[31] ElEconomista, “EEUU fomenta la programación informática con Obama de alumno estrella.” [Online]. Available: http://www.eleconomista.es/tecnologia/noticias/6311475/12/14/EEUU-fomenta-el-aprendizaje-de-programacion-informatica-con-Obama-de-alumno-estrella.html. [Accessed: 21-May-2017].

[32] R. C. White Jr, "Cloud Computing: advantages and disadvantages," *Internet source*, 2010.

[33] B. Kleyman, "Security Breaches, Data Loss, Outages: The Bad Side of Cloud." [Online]. Available: http://www.datacenterknowledge.com/archives/2015/03/16/security-breaches-data-loss-outages-the-bad-side-of-cloud/. [Accessed: 21-May-2017].

[34] T. W. Bynum, "Norbert Wiener and the rise of information ethics," *Inf. Technol. moral Philos.*, pp. 8–25, 2008.

[35] Akamai, "LA BOTNET MIRAI, RESPONSABLE DE LOS ATAQUES MÁS POTENTES Y DEVASTADORES." [Online]. Available: http://seguridadcloud.cso.computerworld.es/whitepapers/content-download/1647b5c3-e841-4852-ba90-04a2376340e8/akamai-mirai-botnet-and-attacks-against-dns-servers-white-paper.pdf. [Accessed: 25-May-2017].

[36] ElPais, "Europol confirma el número de víctimas y países afectados por el incidente de 'WannaCry.'" [Online]. Available: http://internacional.elpais.com/internacional/2017/05/14/actualidad/1494758068_707857.html. [Accessed: 25-May-2017].

[37] CNET, "WhatsApp ya cifra las conversaciones de sus 1.000 millones de usuarios." [Online]. Available: https://www.cnet.com/es/noticias/whatsapp-cifrado-conversaciones-1000-millones-de-usuarios/. [Accessed: 25-May-2017].

[38] Xataka, "Vulnerabilidades de seguridad en SmartTVs y las consecuencias en tu privacidad." [Online]. Available: https://www.xataka.com/televisores/vulnerabilidades-de-seguridad-en-smarttvs-y-las-consecuencias-en-tu-privacidad. [Accessed: 26-May-2017].

[39] R. L. Krutz and R. D. Vines, *Cloud security: A comprehensive guide to secure cloud computing*. Wiley Publishing, 2010.

[40] Geektopia, "Descubierta una base de datos con los datos de 560 millones de usuarios de varios servicios." [Online]. Available: https://www.geektopia.es/es/technology/2017/05/17/noticias/descubierta-una-base-de-datos-con-los-datos-de-560-millones-usuarios-de-varios-servicios.html. [Accessed: 26-May-2017].

[41] Hipertextual, “Vuelven a hackear a Yahoo!: 32 millones de cuentas en peligro.” [Online]. Available: https://hipertextual.com/2017/03/hackeo-de-yahoo. [Accessed: 25-May-2017].

[42] Wired, “Hackers Claim to Auction Data They Stole From NSA-Linked Spies.” [Online]. Available: https://www.wired.com/2016/08/hackers-claim-auction-data-stolen-nsa-linked-spies/. [Accessed: 25-May-2017].

[43] TheVerge, “Google boosts bonuses for finding Chrome bugs by $1,000 or more as participation declines.” [Online]. Available: https://www.theverge.com/2012/8/15/3244812/google-chromium-project-bug-report-bonuses. [Accessed: 26-May-2017].

[44] Quartz, “Today we’re worried about smart TVs, but in the 1980s Russian spies were hacking typewriters.” [Online]. Available: https://qz.com/932448/forget-smart-tvs-in-the-1980s-spies-were-hacking-typewriters/. [Accessed: 26-May-2017].

[45] A. Valdes and K. Skinner, “Adaptive, model-based monitoring for cyber attack detection,” in *International Workshop on Recent Advances in Intrusion Detection*, 2000, pp. 80–93.

[46] J. R. Vacca, *Network and system security*. Elsevier, 2013.

[47] M. Labs, “Predicciones sobre amenazas para 2016.” [Online]. Available: https://mcafee.app.box.com/v/2016predictions. [Accessed: 27-May-2017].

[48] J. A. Lewis, *Assessing the risks of cyber terrorism, cyber war and other cyber threats*. Center for Strategic & International Studies Washington, DC, 2002.

[49] Xataka, “La NSA confirma que Rusia hackeó la ‘infraestructura’ de las elecciones francesas.” [Online]. Available: https://www.xataka.com/privacidad/la-nsa-confirma-que-rusia-hackeo-la-infraestructura-de-las-elecciones-francesas. [Accessed: 26-May-2017].

[50] Xataka, “Hay un arma mas peligrosa de lo que parece: el hackeo de dispositivos médicos.” [Online]. Available: https://www.xataka.com/medicina-y-salud/el-hackeo-de-dispositivos-medicos-va-mucho-mas-alla-de-la-ficcion-de-homeland. [Accessed: 26-May-2017].

[51] FCW, “Rogers: ‘cyber war’ is here to stay.” [Online]. Available: https://fcw.com/articles/2017/05/09/cyber-war-is-here-to-stay.aspx. [Accessed: 26-May-

2017].

[52] K. Popović and Ž. Hocenski, "Cloud computing security issues and challenges," in *MIPRO, 2010 proceedings of the 33rd international convention*, 2010, pp. 344–349.

[53] Cisco, "Informe anual de seguridad 2016." [Online]. Available: http://www.cisco.com/c/m/es_mx/offers/sc04/2016-annual-security-report/index.html?keyCode=001094168#. [Accessed: 27-May-2017].

[54] ITESO, "Educación: La principal herramienta de la seguridad informática." [Online]. Available: http://www.iteso.mx/web/general/detalle?group_id=2303914. [Accessed: 27-May-2017].

[55] Iefangel, "Importancia de la seguridad informática para estudiantes, Padres de Familia y docentes." [Online]. Available: http://iefangel.org/2012/09/25/importancia-de-la-seguridad-informatica-para-estudiantes-padres-de-familia-y-docentes/. [Accessed: 27-May-2017].

[56] TecnologíaEmpresa, "¿Qué seguridad le exiges a tu proveedor de servicios cloud?" [Online]. Available: http://tecnologiaparatuempresa.ituser.es/seguridad/2016/11/que-seguridad-le-exiges-a-tu-proveedor-de-servicios-cloud. [Accessed: 27-May-2017].

[57] RadioAngulo, "Internet, la era de la hiperconexión." .

[58] Cl. ComPUtING, "Cloud computing privacy concerns on our doorstep," *Commun. ACM*, vol. 54, no. 1, pp. 36–38, 2011.

[59] WhatsApp, "Información legal de WhatsApp." [Online]. Available: https://www.whatsapp.com/legal/#key-updates. [Accessed: 28-May-2017].

[60] Gizmodo, "Todo lo que hay que desactivar en Windows 10 para proteger tu privacidad." [Online]. Available: http://es.gizmodo.com/todo-lo-que-hay-que-desactivar-en-windows-10-para-prote-1722426744. [Accessed: 28-May-2017].

[61] ElMundo, "Google admite que no se puede esperar total privacidad al usar Gmail." [Online]. Available: http://www.elmundo.es/elmundo/2013/08/15/navegante/1376537052.html. [Accessed: 28-May-2017].

[62] ElPais, "Google dejará de leer tu correo para poner publicidad." [Online]. Available: http://tecnologia.elpais.com/tecnologia/2017/06/24/actualidad/1498288084_429853.html. [Accessed: 31-May-2017].

[63] Geektopia, "WhatsApp compartirá información de sus usuarios con Facebook para mejorar la publicidad." [Online]. Available: https://www.geektopia.es/es/technology/2016/08/25/noticias/whatsapp-compartira-informacion-de-sus-usuarios-con-facebook-para-mejorar-la-publicidad.html. [Accessed: 28-May-2017].

[64] N. Kshetri, "Privacy and security issues in cloud computing: The role of institutions and institutional evolution," *Telecomm. Policy*, vol. 37, no. 4, pp. 372–386, 2013.

[65] Engadget, "European regulators push Facebook to tighten user privacy rules." [Online]. Available: https://www.engadget.com/2017/05/16/european-regulators-fine-facebook-tighter-user-privacy-rules/. [Accessed: 28-May-2017].

[66] U. Somani, K. Lakhani, and M. Mundra, "Implementing digital signature with RSA encryption algorithm to enhance the Data Security of cloud in Cloud Computing," in *Parallel Distributed and Grid Computing (PDGC), 2010 1st International Conference on*, 2010, pp. 211–216.

[67] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Secur. Priv.*, vol. 8, no. 6, pp. 24–31, 2010.

[68] Tresorit, "What is Zero-Knowledge Encryption?" [Online]. Available: https://tresorit.com/blog/zero-knowledge-encryption/. [Accessed: 28-May-2017].

[69] ITLinuxBlog, "Cloud Computing: Alternativas Open Source." [Online]. Available: http://blog.itlinux.cl/blog/2013/08/14/open-source-y-tecnologias-de-cloud-computing/. [Accessed: 28-May-2017].

[70] A. Whitten and J. D. Tygar, "Why Johnny Can'T Encrypt: A Usability Evaluation of PGP 5.0," in *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, 1999, p. 14.

[71] ElFuturoEsOne, "El creador de Tor te explica cómo navegar con seguridad." [Online].

Available: https://www.youtube.com/watch?v=tEJsKW8AFxs. [Accessed: 28-May-2017].

[72] Google, "Privacidad y Condiciones." [Online]. Available: https://www.google.es/intl/es/policies/privacy/. [Accessed: 28-May-2017].

[73] ElMundo, "¿Cómo me afectan los cambios de Google?" [Online]. Available: http://www.elmundo.es/elmundo/2012/01/25/navegante/1327486750.html. [Accessed: 28-May-2017].

[74] Motherboard, "Hacked Toy Company VTech's TOS Now Says It's Not Liable for Hacks." [Online]. Available: https://motherboard.vice.com/en_us/article/hacked-toy-company-vtech-tos-now-says-its-not-liable-for-hacks. [Accessed: 28-May-2017].

[75] Google, "Google Calendar." [Online]. Available: https://www.google.com/calendar. [Accessed: 29-May-2017].

[76] Microsoft, "Outlook Calendar." [Online]. Available: https://office.live.com/start/Calendar.aspx. [Accessed: 29-May-2017].

[77] Apple, "iCal." [Online]. Available: https://www.apple.com/es/support/ical/index.html. [Accessed: 29-May-2017].

[78] T. Sanamrad, P. Nick, D. Widmer, D. Kossmann, and L. Braun, "My Private Google Calendar and GMail.," *IEEE Data Eng. Bull.*, vol. 35, no. 4, pp. 83–92, 2012.

[79] Open Whisper Systems, "Flock, Private Contact and Calendar Cloud Sync." [Online]. Available: https://whispersystems.org/blog/flock/. [Accessed: 29-May-2017].

[80] Wikipedia, "Signal (software)." [Online]. Available: https://es.wikipedia.org/wiki/Signal_(software). [Accessed: 29-May-2017].

[81] D. Arroyo, J. Diaz, and V. Gayoso, "On the Difficult Tradeoff Between Security and Privacy: Challenges for the Management of Digital Identities," in *International Joint Conference*, 2015, pp. 455–462.

[82] M. D. Network, "Tecnología Web para desarrolladores." [Online]. Available: https://developer.mozilla.org/es/docs/Web. [Accessed: 30-May-2017].

[83] M. D. Network, "HTML5." [Online]. Available:

https://developer.mozilla.org/es/docs/HTML/HTML5. [Accessed: 30-May-2017].

[84] M. D. Network, "CSS3." [Online]. Available: https://developer.mozilla.org/es/docs/Web/CSS/CSS3. [Accessed: 30-May-2017].

[85] JQuery, "jQuery." [Online]. Available: https://jquery.com/. [Accessed: 30-May-2017].

[86] Jq. UI, "jQuery UI." [Online]. Available: https://jqueryui.com/. [Accessed: 30-May-2017].

[87] M. S. Mikowski and J. C. Powell, "Single page web applications," *B W*, 2013.

[88] Node.js, "node.js." [Online]. Available: https://nodejs.org/es/. [Accessed: 30-May-2017].

[89] Electron, "Electron." [Online]. Available: https://electron.atom.io/. [Accessed: 30-May-2017].

[90] RevistaDigitalINESEM, "Tipos de nubes en Cloud Computing." [Online]. Available: https://revistadigital.inesem.es/informatica-y-tics/cloud-computing-tipos-de-nubes/. [Accessed: 29-May-2017].

[91] TheBalance, "Best Cloud Storage Services For Backup." [Online]. Available: https://www.thebalance.com/free-cloud-storage-1356638. [Accessed: 29-May-2017].

[92] Wikipedia, "Web hosting service." [Online]. Available: https://en.wikipedia.org/wiki/Web_hosting_service. [Accessed: 29-May-2017].

[93] Google, "Firebase." [Online]. Available: https://firebase.google.com/?hl=es-419. [Accessed: 29-May-2017].

[94] GitHub, "NeDB." [Online]. Available: https://github.com/louischatriot/nedb. [Accessed: 29-May-2017].

[95] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Annual International Cryptology Conference*, 1999, pp. 537–554.

[96] Wikipedia, "Cifrado (criptografía)." [Online]. Available: https://es.wikipedia.org/wiki/Cifrado_(criptografía). [Accessed: 30-May-2017].

[97] S. Mrdovic and B. Perunicic, "Kerckhoffs' principle for intrusion detection," in *Telecommunications Network Strategy and Planning Symposium, 2008. Networks 2008.*

The 13th International, 2008, pp. 1–8.

[98] E. Stark, M. Hamburg, and D. Boneh, "Symmetric cryptography in javascript," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, 2009, pp. 373–381.

[99] N. W. Group, "OpenPGP Message Format." [Online]. Available: https://tools.ietf.org/html/rfc4880. [Accessed: 30-May-2017].

[100] OpenPGPjs, "OpenPGP JavaScript." [Online]. Available: https://openpgpjs.org/. [Accessed: 30-May-2017].

[101] N. W. Group, "Internet Calendaring and Scheduling Core Object Specification (iCalendar)." [Online]. Available: https://tools.ietf.org/html/rfc5545. [Accessed: 30-May-2017].

[102] Wikipedia, "iCalendar." [Online]. Available: https://en.wikipedia.org/wiki/ICalendar. [Accessed: 30-May-2017].

[103] Wikipedia, "QR code." [Online]. Available: https://en.wikipedia.org/wiki/QR_code. [Accessed: 30-May-2017].

[104] P. Bourque, R. E. Fairley, and others, *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.

[105] AvePoint, "Privacy and Security by Design: The New Default under GDPR." [Online]. Available: https://www.avepoint.com/blog/avepoint-blog/privacy-and-security-by-design-gdpr/. [Accessed: 30-May-2017].

[106] S. Fischer-Hübner, *IT-security and privacy: design and use of privacy-enhancing security mechanisms*. Springer-Verlag, 2001.

[107] L. Antonenkov, S. Romanovskiy, N. Uraltsev, and A. Prokofiev, "Client-side encryption." Google Patents, 2014.

[108] H. M. Zeidler, "End-to-end encryption system and method of operation." Google Patents, 1986.

[109] W. De Groef, "Client-and Server-Side Security Technologies for JavaScript Web Applications," 2016.

[110] M. Fowler, "Serverless Architectures." [Online]. Available: https://martinfowler.com/articles/serverless.html. [Accessed: 30-May-2017].

[111] A. Wiggins, "The Twelve-Factor App." [Online]. Available: https://12factor.net/es/. [Accessed: 30-May-2017].

[112] U. Europea, "EU General Data Protection Regulation (GDPR)." [Online]. Available: http://www.eugdpr.org/. [Accessed: 30-May-2017].

[113] S. Barocas and H. Nissenbaum, "Big data's end run around procedural privacy protections," *Commun. ACM*, vol. 57, no. 11, pp. 31–33, 2014.

[114] Yubico, "How to Use Your YubiKey With OpenPGP." [Online]. Available: https://www.yubico.com/support/knowledge-base/categories/articles/use-yubikey-openpgp/. [Accessed: 31-May-2017].

[115] A. Sánchez Gómez, "Development of a software infrastructure for the secure distribution of documents using free cloud storage," *Trabajo Fin de Máster, Escuela Politécnica Superior, Universidad Autónoma de Madrid*, 2016. [Online]. Available: https://repositorio.uam.es/handle/10486/675887.

[116] J. Diaz, D. Arroyo, and F. B. Rodriguez, "A formal methodology for integral security design and verification of network protocols," *J. Syst. Softw.*

[117] J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[118] K. Radke, C. Boyd, J. G. Nieto, and H. Bartlett, "CHURNs: freshness assurance for humans," *Comput. J.*, p. bxu073, 2014.